TOWARDS OUTLIER DETECTION FOR HIGH-DIMENSIONAL DATA STREAMS USING PROJECTED OUTLIER ANALYSIS STRATEGY

by

Ji Zhang

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
December 2008

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "TOWARDS OUTLIER DETECTION FOR HIGH-DIMENSIONAL DATA STREAMS USING PROJECTED OUTLIER ANALYSIS STRATEGY" by Ji Zhang in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: December 10, 2008

External Examiner: _____

Research Supervisor: _____

Examining Committee: _____

_____

_____

_____

DALHOUSIE UNIVERSITY

DATE: December 10, 2008

AUTHOR: Ji Zhang

TITLE: TOWARDS OUTLIER DETECTION FOR HIGH-DIMENSIONAL DATA STREAMS USING PROJECTED OUTLIER ANALYSIS STRATEGY

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: PhD CONVOCATION: May YEAR: 2009

Table of Contents

List of Tables

List of Figures

Abstract

Outlier detection is an important research problem in data mining that aims to discover useful abnormal and irregular patterns hidden in large data sets. Most existing outlier detection methods only deal with static data with relatively low dimensionality. Recently, outlier detection for high-dimensional stream data became a new emerging research problem. A key observation that motivates this research is that outliers in high-dimensional data are projected outliers, *i.e.*, they are embedded in lower-dimensional subspaces. Detecting projected outliers from high-dimensional stream data is a very challenging task for several reasons. First, detecting projected outliers is difficult even for high-dimensional static data. The exhaustive search for the outlying subspaces where projected outliers are embedded is a NP problem. Second, the algorithms for handling data streams are constrained to take only one pass to process the streaming data with the conditions of space limitation and time criticality. The currently existing methods for outlier detection are found to be ineffective for detecting projected outliers in high-dimensional data streams.

In this thesis, we present a new technique, called the Stream Project Outlier deTector (SPOT), which attempts to detect projected outliers in high-dimensional data streams. SPOT employs an innovative window-based time model in capturing dynamic statistics from stream data, and a novel data structure containing a set of top sparse subspaces to detect projected outliers effectively. SPOT also employs a multi-objective genetic algorithm as an effective search method for finding the outlying subspaces where most projected outliers are embedded. The experimental results demonstrate that SPOT is efficient and effective in detecting projected outliers for high-dimensional data streams. The main contribution of this thesis is that it provides a backbone in tackling the challenging problem of outlier detection for high-dimensional data streams. SPOT can facilitate the discovery of useful abnormal patterns and can be potentially applied to a variety of high demand applications, such as for sensor network data monitoring, online transaction protection, etc.

List of Abbreviations Used

SPOT: Stream Projected Outlier Detector

SST: Sparse Subspace Template

FS: Fixed Subspaces

US: Unsupervised Subspaces

SS: Supervised Subspaces

BCS: Base Cell Summary

PCS: Projected Cell Summary

RD: Relative Density

IRSD: Inverse Relative Standard Deviation

IkRD: Inverse k-Relative Distance

MOGA: Multiobjective Genetic Algorithm

CLT: Crossover Lookup Table

MTPT: Mutation Transition Probability Table

MSE: Mean Square Error

SC: Silhouette Coefficient

ROC: Receiver Operating Characteristic

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Qigang Gao and Dr. Hai Wang for their dedicated supervision during my Ph.D study. Their endless help, care, kindness, patience, generosity, and thoughtful considerations are greatly valued.

I would like to thank Dr. Malcolm Heywood for his wonderful course on Genetic Algorithms. It is his course that has stimulated much of my interest in this area, which greatly contributes to my Ph.D research.

I greatly appreciate Dr. John McHugh for his willingness to share his wide scope of knowledge with me and to give valuable suggestions on some parts of my Ph.D research.

I would also like to thank Dr. Christian Blouin for his interesting course in Bioinformatics. I learned much from this course that paved a good foundation for my long-term research career development.

I would like to thank Killam Trust as well for awarding me the prestigious Killam Predoctoral Scholarship, which provides a strong financial support to my Ph.D research activities. I deeply appreciate Dr. Qigang Gao, Dr. Hai Wang and Dr. Malcolm Heywood for their unreserved support in my application for this scholarship.

Thanks also go to the faculty of Computer Science for the good research facility and atmosphere it created. In particular, I am very grateful to the personal help from Dr. Srinivas Sampalli and Ms. Menen Teferra for many times.

I would like to thank my all colleagues and friends for the good time we have had in Dalhousie. I would like to thank my family for their continued support and care. It would be impossible to finish my Ph.D study without their continuous understanding and support.

Chapter 1

Introduction

Outlier detection is an important research problem in data mining that aims to find objects that are considerably dissimilar, exceptional and inconsistent with respect to the majority data in an input database [60]. In recent years, we have witnessed a tremendous research interest sparked by the explosion of data collected and transferred in the format of streams. This poses new opportunities as well as challenges for research efforts in outlier detection. A data stream is a real-time, continuous and ordered (implicitly by arrival sequence or explicitly by timestamp) sequence of items. Examples of data streams include network traffic, telecommunications data, financial market data, data from sensors that monitor the weather and environment, surveillance video and so on. Outlier detection from stream data can find items (objects or points) that are abnormal or irregular with respect to the majority of items in the whole or a horizon/window of the stream. Outlier detection in data streams can be useful in many fields such as analysis and monitoring of network traffic data (e.g., connection-oriented records), web log, wireless sensor networks and financial transactions, etc.

A key observation that motivates this research is that outliers existing in high-dimensional data streams are embedded in some lower-dimensional subspaces. Here, a subspace refers to as the data space consisting of a subset of attributes. These outliers are termed *projected outliers* in the high-dimensional space. The existence of projected outliers is due to the fact that, as the dimensionality of data goes up, data tend to become equally distant from each other. As a result, the difference of data points' outlier-ness will become increasingly weak and thus undistinguishable. Only in moderate or low dimensional subspaces can significant outlier-ness of data be observed. This phenomena is commonly referred to as the *curse of dimensionality*

[17]. Because most of state-of-the-art outlier detection methods perform detection in the full data space, thus the projected outliers cannot be found by these methods. This will lead to a loss of interesting and potentially useful abnormal patterns hidden in high-dimensional data streams.

In this research, we will study the problem of detecting projected outliers from high-dimensional data streams. This problem can be formulated as follows: given a data stream $\mathcal{D}$ with a potentially unbounded size of φ-dimensional data points, for each data point $p_i = \{p_{i1}, p_{i2}, \ldots, p_{i\varphi}\}$ in $\mathcal{D}$, projected outlier detection method performs a mapping as

$$f : p_i \rightarrow (b, S_i, Score_i)$$

where each data p_i is mapped to a triplet $(b, S_i, Score_i)$. b is a Boolean variable indicating whether or not p_i is a projected outlier. If p_i is a projected outlier (*i.e.*, $b = true$), then S_i is the set of outlying subspaces of p_i and $Score_i$ is the corresponding outlier-ness score of p_i in each subspace of S_i. In the case that p_i is a normal data, we have $b = false$, $S_i = \emptyset$ and $Score_i$ is not applicable.

The results of the detection method will be a set of projected outliers and their associated outlying subspace(s) and outlier-ness score to characterize the context and strength of the projected outliers detected. The results, denoted by $\mathcal{A}$, can be formally expressed as

$$\mathcal{A} = \{< o, S, Score >, \text{o} \in O\}$$

where O denotes the set of projected outliers detected. The users have the discretion to pick up the top k projected outliers that have the highest outlier-ness from O. In contrast, the traditional definition of outliers does not explicitly present outlying subspaces of outliers in the final result as outliers are detected in the full or a pre-specified data space that is known to users before outliers are detected.

Detecting projected outliers in high-dimensional data streams is a nontrivial research problem. Two major challenges we face in tackling this problem are elaborated as follows:

1. First, finding the right outlying subspaces for projected outliers is crucial to detection performance of the algorithm. Once these outlying subspaces have been found, detecting projected outliers in these subspace will then become a much easier task. Nevertheless, the number of possible subspaces increases dramatically with the data dimensionality. Thus, finding the outlying subspaces of the data through an exhaustive search of the space lattice is rather computationally demanding and totally infeasible when the dimensionality of data is high. In light of this, the outlier detection algorithm should be reasonably efficient to find the right subspaces in which projected outliers can be accurately detected;

2. Another aspect of the challenge originates from the characteristics of streaming data themselves. First, data streams can only be accessed in the order of their arrivals and random access is disallowed. Second, data streams are potentially unbound in size and the space available to store information is supposed to be small. Finally, data objects in the stream usually have implicit or explicit time concept (*e.g.*, timestamps). Because of these unique features of data streams, data stream outlier detection algorithms can only have one pass over data streams and process data in an incremental, online and real-time paradigm. In addition, they should feature constant and short time for processing each data object and limited space for storing information. They need to employ dynamic and space-economic data synopsis that can be updated incrementally and efficiently. Finally, they are expected to take into account the time concept in the detection process. They should apply appropriate time model(s) to discriminate data arriving at different time, and have necessary capability to cope with concept drift that may occur in the streams.

There have been intensive research efforts in outlier detection in the past decade. The existing methods can be classified based on different criteria. Specifically, they can be classified based upon whether they are used for low-dimensional or high-dimensional data, whether they use the full set of attributes for outlier detection or detect outliers in subspaces, and whether they can only handle static data or they can deal with stream data. Most of the conventional outlier detection techniques are

only applicable to relatively low dimensional static data [26][76][77][101][111]. Because they use the full set of attributes for outlier detection, thus they are not able to detect projected outliers. They cannot handle data streams either. Recently, there are some emerging work in dealing with outlier detection either in high-dimensional static data or data streams. However, there has not been any reported concrete research work so far for exploring the intersection of these two active research directions. For those methods in projected outlier detection in high-dimensional space [14][123][131][126][128], they can detect projected outliers that are embedded in subspaces. However, their measurements used for evaluating points' outlier-ness are not incrementally updatable and many of the methods involve multiple scans of data, making them incapable of handling data streams. For instance, [14][123] use the Sparsity Coefficient to measure data sparsity. Sparsity Coefficient is based on an equi-depth data partition that has to be updated frequently from the data stream. This will be expensive and such updates will require multiple scans of data. [131][126][128] use data sparsity metrics that are based on distance involving the concept of k nearest neighbors (kNN). This is not suitable for data streams either as one scan of data is not sufficient for retaining kNN information of data points. One the other hand, the techniques for tackling outlier detection in data streams [100][1] rely on full data space to detect outliers and thus projected outliers cannot be discovered by these techniques. As such, it is desirable to propose a new method that well solves the drawbacks of these existing methods.

In this thesis, we present a new technique, called Stream Projected Outlier deTector (SPOT), to approach the problem of outlier detection in high-dimensional data streams. The major contributions of this research can be summarized as follows:

- In SPOT, we employ a new window-based time model and decaying data summaries to capture statistics from the data streams for outlier detection. The time model is able to approximate the conventional window-based model without maintaining the detailed data in the window or keeping multiple snapshots of data synopsis. The decaying data summaries can be efficiently computed and incrementally maintained, enabling SPOT to handle fast data streams;

- SPOT constructs a Sparse Subspace Template (SST) to detect projected outliers. SST consists of a number of mutually supplemented subspace groups that contribute collectively to an effective detection of projected outliers. SPOT is able to perform supervised and/or unsupervised learning to construct SST, providing a maximum level of flexibility to users. Self-evolution of SST has also been incorporated into SPOT to greatly enhance its adaptability to dynamics of data streams;

- Unlike most of other outlier detection methods that measure outlier-ness of data points based on a single criterion, SPOT adopts a more flexible framework of using multiple measurements for this purpose. SPOT utilizes the Multi-Objective Genetic Algorithm (MOGA) as an effective search method to find subspaces that are able to optimize all the criteria for constructing SST;

- Last but not the least, we show that SPOT is efficient and effective in detecting projected outliers in subspaces and outperforms the major existing methods through experiments on both synthetic and real-life data streams.

Roadmap

The reminder of this thesis is organized as follows. Chapter 2 will present a review on the existing methods for outlier detection. The basics of SPOT, including the time model, data synopsis and definition of projected outliers, etc, will be elaborated in Chapter 3. In Chapter 4, we dwell on algorithms of SPOT, with emphasis on the learning and detection stages of SPOT. The Multi-Objective Genetic Algorithm (MOGA), used to find outlying subspaces of streaming data for constructing SST, is discussed in Chapter 5. We, in Chapter 6, report performance evaluation of SPOT. The final chapter concludes this thesis.

Chapter 2

Related Work

We have witnessed considerable research efforts in outlier detection in the past a few years. This section presents a review on the major state-of-the-art outlier detection methods. To facilitate a systematic survey of the existing outlier detection methods, the scope of this review is first clearly specified. The organization of the literature review is as follows. We will first review the conventional outlier detection techniques that are primarily suitable for relatively low-dimensional static data, followed by some of recent advancements in outlier detection for high-dimensional static data and data streams.

2.1 Scope of the Review

Before the review of outlier detection methods is presented, it is necessary for us to first explicitly specify the scope of this review. There have been a lot of research work in detecting different kinds of outliers from various types of data where the techniques outlier detection methods utilize differ considerably. Most of the existing outlier detection methods detect the so-called *point outliers from vector-like data sets*. This is the focus of this review as well as of this thesis. Another common category of outliers that has been investigated is called *collective outliers*. Besides the vector-like data, outliers can also be detected from other types of data such as sequences, trajectories and graphs, etc. In the reminder of this subsection, we will discuss briefly different types of outliers.

First, outliers can be classified as point outliers and collective outliers based on the number of data instances involved in the concept of outliers.

- **Point outliers.** In a given set of data instances, an individual outlying instance is termed as a point outlier. This is the simplest type of outliers and is the focus

of majority of existing outlier detection schemes [33]. A data point is detected as a point outlier because it displays outlier-ness at its own right, rather than together with other data points. In most cases, data are represented in vectors as in the relational databases. Each tuple contains a specific number of attributes. The principled method for detecting point outliers from vector-type data sets is to quantify, through some outlier-ness metrics, the extent to which each single data is deviated from the other data in the data set.

- **Collective outliers.** A collective outlier represents a collection of data instances that is outlying with respect to the entire data set. The individual data instance in a collective outlier may not be outlier by itself, but the joint occurrence as a collection is anomalous [33]. Usually, the data instances in a collective outlier are related to each other. A typical type of collective outliers are sequence outliers, where the data are in the format of an ordered sequence.

Outliers can also be categorized into vector outliers, sequence outliers, trajectory outliers and graph outliers, etc, depending on the types of data from where outliers can be detected.

- **Vector outliers.** Vector outliers are detected from vector-like representation of data such as the relational databases. The data are presented in tuples and each tuple has a set of associated attributes. The data set can contain only numeric attributes, or categorical attributes or both. Based on the number of attributes, the data set can be broadly classified as low-dimensional data and high-dimensional data, even though there is not a clear cutoff between these two types of data sets. As relational databases still represent the mainstream approaches for data storage, therefore, vector outliers are the most common type of outliers we are dealing with.

- **Sequence outliers.** In many applications, data are presented as a sequence. A good example of a sequence database is the computer system call log where the computer commands executed, in a certain order, are stored. A sequence

of commands in this log may look like the following sequence: *http-web, buffer-overflow, http-web, http-web, smtp-mail, ftp, http-web, ssh.* Outlying sequence of commands may indicate a malicious behavior that potentially compromises system security. In order to detect abnormal command sequences, normal command sequences are maintained and those sequences that do not match any normal sequences are labeled sequence outliers. Sequence outliers are a form of collective outlier.

- **Trajectory outliers.** Recent improvements in satellites and tracking facilities have made it possible to collect a huge amount of trajectory data of moving objects. Examples include vehicle positioning data, hurricane tracking data, and animal movement data [83]. Unlike a vector or a sequence, a trajectory is typically represented by a set of key features for its movement, including the coordinates of the starting and ending points; the average, minimum, and maximum values of the directional vector; and the average, minimum, and maximum velocities. Based on this representation, a weighted-sum distance function can be defined to compute the difference of trajectory based on the key features for the trajectory [78]. A more recent work proposed a partition-and-detect framework for detecting trajectory outliers [83]. The idea of this method is that it partitions the whole trajectory into line segments and tries to detect outlying line segments, rather than the whole trajectory. Trajectory outliers can be point outliers if we consider each single trajectory as the basic data unit in the outlier detection. However, if the moving objects in the trajectory are considered, then an abnormal sequence of such moving objects (constituting the sub-trajectory) is a collective outlier.

- **Graph outliers.** Graph outliers represent those graph entities that are abnormal when compared with their peers. The graph entities that can become outliers include nodes, edges and sub-graphs. For example, Sun *et al.* investigate the detection of anomalous nodes in a bipartite graph [107][108]. Autopart detects outlier edges in a general graph [32]. Noble *et al.* study anomaly detection on a general graph with labeled nodes and try to identify abnormal

substructure in the graph [90]. Graph outliers can be either point outliers (*e.g.*, node and edge outliers) or collective outliers (*e.g.*, sub-graph outliers).

Unless otherwise stated, all the outlier detection methods discussed in this review refer to those methods for detecting point outliers from vector-like data sets.

2.2 Outlier Detection Methods for Low Dimensional Data

The earlier research work in outlier detection mainly deals with static datasets with relatively low dimensions. Literature on these work can be broadly classified into four major categories based on the techniques they used, *i.e.*, statistical methods, distance-based methods, density-based methods and clustering-based methods.

2.2.1 Statistical Detection Methods

Statistical outlier detection methods [28, 57] rely on the statistical approaches that assume a distribution or probability model to fit the given dataset. Under the distribution assumed to fit the dataset, the outliers are those points that do not agree with or conform to the underlying model of the data.

The statistical outlier detection methods can be broadly classified into two categories, *i.e.*, the parametric methods and the non-parametric methods. The major differences between these two classes of methods lie in that the parametric methods assume the underlying distribution of the given data and estimate the parameters of the distribution model from the given data [41] while the non-parametric methods do not assume any knowledge of distribution characteristics [38].

Statistical outlier detection methods (parametric and non-parametric) typically take two stages for detecting outliers, *i.e.*, the training stage and test stage.

- **Training stage**. The training stage mainly involves fitting a statistical model or building data profiles based on the given data. Statistical techniques can be performed in a supervised, semi-supervised, and unsupervised manner. Supervised techniques estimate the probability density for normal instances and outliers. Semi-supervised techniques estimate the probability density for either

normal instances, or outliers, depending on the availability of labels. Unsupervised techniques determine a statistical model or profile which fits all or the majority of the instances in the given data set;

- **Test stage**. Once the probabilistic model or profile is constructed, the next step is to determine if a given data instance is an outlier with respect to the model/profile or not. This involves computing the posterior probability of the test instance to be generated by the constructed model or the deviation from the constructed data profile. For example, we can find the distance of the data instance from the estimated mean and declare any point above a threshold to be an outlier [51].

Parametric Methods

Parametric statistical outlier detection methods explicitly assume the probabilistic or distribution model(s) for the given data set. Model parameters can be estimated using the training data based upon the distribution assumption. The major parametric outlier detection methods include Gaussian model-based and regression model-based methods.

A. Gaussian Models

Detecting outliers based on Gaussian distribution models have been intensively studied. The training stage typically performs estimation of the mean and variance (or standard deviation) of the Gaussian distribution using Maximum Likelihood Estimates (MLE). To ensure that the distribution assumed by human users is the optimal or close-to-optima underlying distribution the data fit, statistical discordany tests are normally conducted in the test stage [28][18][21]. So far, over one hundred discordancy/outlier tests have been developed for different circumstances, depending on the parameter of dataset (such as the assumed data distribution) and parameter of distribution (such as mean and variance), and the expected number of outliers [60][76]. The rationale is that some small portion of points that have small probability of occurrence in the population are identified as outliers. The commonly used outlier tests for normal distributions are the *mean-variance test* and *box-plot test* [84][59][106][53].

In the mean-variance test for a Gaussian distribution $N(\mu, \sigma^2)$, where the population has a mean μ and variance σ, outliers can be considered to be points that lie 3 or more standard deviations (*i.e.*, $\geq 3\sigma$) away from the mean [50]. This test is general and can be applied to some other commonly used distributions such as Student t-distribution and Poisson distribution, which feature a fatter tail and a longer right tail than a normal distribution, respectively. The box-plot test draws on the box plot to graphically depict the distribution of data using five major attributes, *i.e.*, smallest non-outlier observation (min), lower quartile (Q1), median, upper quartile (Q3), and largest non-outlier observation (max). The quantity Q3-Q1 is called the *Inter Quartile Range (IQR)*. IQR provides a means to indicate the boundary beyond which the data will be labeled as outliers; a data instance will be labeled as an outlier if it is located 1.5*IQR times lower than Q1 or 1.5*IQR times higher than Q3.

In some cases, a mixture of probabilistic models may be used if a single model is not sufficient for the purpose of data modeling. If labeled data are available, two separate models can be constructed, one for the normal data and another for the outliers. The membership probability of the new instances can be quantified and they are labeled as outliers if their membership probability of outlier probability model is higher than that of the model of the normal data. The mixture of probabilistic models can also be applied to unlabeled data, that is, the whole training data are modeled using a mixture of models. A test instance is considered to be an outlier if it is found that it does not belong to any of the constructed models.

B. Regression Models

If the probabilistic model is unknown regression can be employed for model construction. The regression analysis aims to find a dependence of one/more random variable(s) $\mathcal{Y}$ on another one/more variable(s) $\mathcal{X}$. This involves examining the conditional probability distribution $\mathcal{Y}|\mathcal{X}$. Outlier detection using regression techniques are intensively applied to time-series data [4][2][46][1][82]. The training stage involves constructing a regression model that fits the data. The regression model can either be a linear or non-linear model, depending on the choice from users. The test stage tests the regression model by evaluating each data instance against the model. More

specifically, such test involves comparing the actual instance value and its projected value produced by the regression model. A data point is labeled as an outlier if a remarkable deviation occurs between the actual value and its expected value produced by the regression model.

Basically speaking, there are two ways to use the data in the dataset for building the regression model for outlier detection, namely the *reverse search* and *direct search* methods. The reverse search method constructs the regression model by using all data available and then the data with the greatest error are considered as outliers and excluded from the model. The direct search approach constructs a model based on a portion of data and then adds new data points incrementally when the preliminary model construction has been finished. Then, the model is extended by adding most fitting data, which are those objects in the rest of the population that have the least deviations from the model constructed thus far. The data added to the model in the last round, considered to be the least fitting data, are regarded to be outliers.

Non-parametric Methods

The outlier detection techniques in this category do not make any assumptions about the statistical distribution of the data. The most popular approaches for outlier detection in this category are histograms and Kernel density function methods.

A. Histograms

The most popular non-parametric statistical technique is to use histograms to maintain a profile of data. Histogram techniques by nature are based on the frequency or counting of data.

The histogram based outlier detection approach is typically applied when the data has a single feature. Mathematically, a histogram for a feature of data consists of a number of disjoint bins (or buckets) and the data are mapped into one (and only one) bin. Represented graphically by the histogram graph, the height of bins corresponds to the number of observations that fall into the bins. Thus, if we let n be the total number of instances, k be the total number of bins and m_i be the number of data point in the i^{th} bin ($1 \leq i \leq k$), the histogram satisfies the following condition $n = \sum_{i=1}^{k} m_i$.

The training stage involves building histograms based on the different values taken by that feature in the training data.

The histogram techniques typically define a measure between a new test instance and the histogram based profile to determine if it is an outlier or not. The measure is defined based on how the histogram is constructed in the first place. Specifically, there are three possible ways for building a histogram:

1. The histogram can be constructed only based on normal data. In this case, the histogram only represents the profile for normal data. The test stage evaluates whether the feature value in the test instance falls in any of the populated bins of the constructed histogram. If not, the test instance is labeled as an outlier [5] [68][58];

2. The histogram can be constructed only based on outliers. As such, the histogram captures the profile for outliers. A test instance that falls into one of the populated bins is labeled as an outlier [39]. Such techniques are particularly popular in intrusion detection community [41][45] [35] and fraud detection [49];

3. The histogram can be constructed based on a mixture of normal data and outliers. This is the typical case where histogram is constructed. Since normal data typically dominate the whole data set, thus the histogram represents an approximated profile of normal data. The sparsity of a bin in the histogram can be defined as the ratio of frequency of this bin against the average frequency of all the bins in the histogram. A bin is considered as sparse if such ratio is lower than a user-specified threshold. All the data instance falling into the sparse bins are labeled as outliers.

The first and second ways for constructing histogram, as presented above, rely on the availability of labeled instances, while the third one does not.

For multivariate data, a common approach is to construct feature-wise histograms. In the test stage, the probability for each feature value of the test data is calculated and then aggregated to generate the so-called *outlier score*. A low probability value corresponds a higher outlier score of that test instance. The aggregation of per-feature

likelihoods for calculating outlier score is typically done using the following equation:

$$Outlier\_Score = \sum_{f \in F} w_f \cdot (1 - p_f)/|F|$$

where w_f denotes the weight assigned for feature f, p_f denotes the probability for the value of feature f and F denotes the set of features of the dataset. Such histogram-based aggregation techniques have been used in intrusion detection in system call data [42], fraud detection [49], damage detection in structures [85] [88] [89], network intrusion detection [115] [117], web-based attack detection [81], Packet Header Anomaly Detection (PHAD), Application Layer Anomaly Detection (ALAD) [87], NIDES (by SRI International) [5] [12] [99]. Also, a substantial amount of research has been done in the field of outlier detection for sequential data (primarily to detect intrusions in computer system call data) using histogram based techniques. These techniques are fundamentally similar to the instance based histogram approaches as described above but are applied to sequential data to detect collective outliers.

Histogram based detection methods are simple to implement and hence are quite popular in domain such as intrusion detection. But one key shortcoming of such techniques for multivariate data is that they are not able to capture the interactions between different attributes. An outlier might have attribute values that are individually very frequent, but their combination is very rare. This shortcoming will become more salient when dimensionality of data is high. A feature-wise histogram technique will not be able to detect such kinds of outliers. Another challenge for such techniques is that users need to determine an optimal size of the bins to construct the histogram.

B. Kernel Functions

Another popular non-parametric approach for outlier detection is the parzen windows estimation due to Parzen [94]. This involves using Kernel functions to approximate the actual density distribution. A new instance which lies in the low probability area of this density is declared to be an outlier.

Formally, if $x_1, x_2, ..., x_N$ are IID (independently and identically distributed) samples of a random variable x, then the Kernel density approximation of its *probability*

density function (pdf) is

$$f_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x - x_i}{h})$$

where K is Kernel function and h is the bandwidth (smoothing parameter). Quite often, K is taken to be a standard Gaussian function with mean $\mu = 0$ and variance $\sigma^2 = 1$:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

Novelty detection using Kernel function is presented by [19] for detecting novelties in oil flow data. A test instance is declared to be novel if it belongs to the low density area of the learnt density function. Similar application of parzen windows is proposed for network intrusion detection [34] and for mammographic image analysis [110]. A semi-supervised probabilistic approach is proposed to detect novelties [38]. Kernel functions are used to estimate the probability distribution function (pdf) for the normal instances. Recently, Kernel functions are used in outlier detection in sensor networks [100][30].

Kernel density estimation of pdf is applicable to both univariate and multivariate data. However, the pdf estimation for multivariate data is much more computationally expensive than the univariate data. This renders the Kernel density estimation methods rather inefficient in outlier detection for high-dimensional data.

Advantages and Disadvantages of Statistical Methods

Statistical outlier detection methods feature some advantages. They are mathematically justified and if a probabilistic model is given, the methods are very efficient and it is possible to reveal the meaning of the outliers found [93]. In addition, the model constructed, often presented in a compact form, makes it possible to detect outliers without storing the original datasets that are usually of large sizes.

However, the statistical outlier detection methods, particularly the parametric methods, suffer from some key drawbacks. First, they are typically not applied in a multi-dimensional scenario because most distribution models typically apply to

the univariate feature space. Thus, they are unsuitable even for moderate multi-dimensional data sets. This greatly limits their applicability as in most practical applications the data is multiple or even high dimensional. In addition, a lack of the prior knowledge regarding the underlying distribution of the dataset makes the distribution-based methods difficult to use in practical applications. A single distribution may not model the entire data because the data may originate from multiple distributions. Finally, the quality of results cannot be guaranteed because they are largely dependent on the distribution chosen to fit the data. It is not guaranteed that the data being examined fit the assumed distribution if there is no estimate of the distribution density based on the empirical data. Constructing such tests for hypothesis verification in complex combinations of distributions is a nontrivial task whatsoever. Even if the model is properly chosen, finding the values of parameters requires complex procedures. From above discussion, we can see the statistical methods are rather limited to large real-world databases which typically have many different fields and it is not easy to characterize the multivariate distribution of exemplars.

For non-parametric statistical methods, such as histogram and Kernal function methods, they do not have the problem of distribution assumption that the parametric methods suffer and they both can deal with data streams containing continuously arriving data. However, they are not appropriate for handling high-dimensional data. Histogram methods are effective for a single feature analysis, but they lose much of their effectiveness for multi or high-dimensional data because they lack the ability to analyze multiple feature simultaneously. This prevents them from detecting subspace outliers. Kernel function methods are appropriate only for relatively low dimensional data as well. When the dimensionality of data is high, the density estimation using Kernel functions becomes rather computationally expensive, making it inappropriate for handling high-dimensional data streams.

2.2.2 Distance-based Methods

There have already been a number of different ways for defining outliers from the perspective of distance-related metrics. Most existing metrics used for distance-based

outlier detection techniques are defined based upon the concepts of *local neighborhood* or k nearest neighbors (kNN) of the data points. The notion of distance-based outliers does not assume any underlying data distributions and generalizes many concepts from distribution-based methods. Moreover, distance-based methods scale better to multi-dimensional space and can be computed much more efficiently than the statistical-based methods.

In distance-based methods, distance between data points is needed to be computed. We can use any of the L_p metrics like the Manhattan distance or Euclidean distance metrics for measuring the distance between a pair of points. Alternately, for some other application domains with presence of categorical data (*e.g.*, text documents), non-metric distance functions can also be used, making the distance-based definition of outliers very general. Data normalization is normally carried out in order to normalize the different scales of data features before outlier detection is performed.

A. Local Neighborhood Methods

The first notion of distance-based outliers, called $DB(k, \lambda)$-Outlier, is due to Knorr and Ng [76]. It is defined as follows. A point p in a data set is a $DB(k, \lambda)$-Outlier, with respect to the parameters k and λ, if no more than k points in the data set are at a distance λ or less (*i.e.*, $\lambda-$neighborhood) from p. This definition of outliers is intuitively simple and straightforward. The major disadvantage of this method, however, is its sensitivity to the parameter λ that is difficult to specify a priori. As we know, when the data dimensionality increases, it becomes increasingly difficult to specify an appropriate circular local neighborhood (delimited by λ) for outlier-ness evaluation of each point since most of the points are likely to lie in a thin shell about any point [24]. Thus, a too small λ will cause the algorithm to detect all points as outliers, whereas no point will be detected as outliers if a too large λ is picked up. In other words, one needs to choose an appropriate λ with a very high degree of accuracy in order to find a modest number of points that can then be defined as outliers.

To facilitate the choice of parameter values, this first local neighborhood distance-based outlier definition is extended and the so-called $DB(pct, d_{min})$-Outlier is proposed which defines an object in a dataset as a $DB(pct, d_{min})$-Outlier if at least

pct% of the objects in the datasets have the distance larger than d_{min} from this object [77][78]. Similar to $DB(k, \lambda)$-Outlier, this method essentially delimits the local neighborhood of data points using the parameter d_{min} and measures the outlierness of a data point based on the percentage, instead of the absolute number, of data points falling into this specified local neighborhood. As pointed out in [74] and [75], $DB(pct, d_{min})$ is quite general and is able to unify the exisiting statisical detection methods using discordancy tests for outlier detection. For exmaple, $DB(pct, d_{min})$ unifies the definition of outliers using a normal distribution-based discordancy test with $pct = 0.9988$ and $d_{min} = 0.13$. The specification of pct is obviously more intuitive and easier than the specification of k in $DB(k, \lambda)$-Outliers [77]. However, $DB(pct, d_{min})$-Outlier suffers a similar problem as $DB(pct, d_{min})$-Outlier in specifying the local neighborhood parameter d_{min}.

To efficiently calculate the number (or percentage) of data points falling into the local neighborhood of each point, three classes of algorithms have been presented, *i.e.*, the nested-loop, index-based and cell-based algorithms. For easy of presentation, these three algorithms are discussed for detecting $DB(k, \lambda)$-Outlier.

The *nested-loop algorithm* uses two nested loops to compute $DB(k, \lambda)$-Outlier. The outer loop considers each point in the dataset while the inner loop computes for each point in the outer loop the number (or percentage) of points in the dataset falling into the specified λ-neighborhood. This algorithm has the advantage that it does not require the indexing structure be constructed at all that may be rather expensive at most of the time, though it has a quadratic complexity with respect to the number of points in the dataset.

The *index-based algorithm* involves calculating the number of points belonging to the λ-neighborhood of each data by intensively using a pre-constructed multi-dimensional index structure such as R^*-tree [27] to facilitate kNN search. The complexity of the algorithm is approximately logarithmic with respect to the number of the data points in the dataset. However, the construction of index structures is sometimes very expensive and the quality of the index structure constructed is not easy to guarantee.

In the *cell-based algorithm*, the data space is partitioned into cells and all the data points are mapped into cells. By means of the cell size that is known a priori, estimates of pair-wise distance of data points are developed, whereby heuristics (pruning properties) are presented to achieve fast outlier detection. It is shown that three passes over the dataset are sufficient for constructing the desired partition. More precisely, the $d-$dimensional space is partitioned into cells with side length of $\frac{\lambda}{2\sqrt{d}}$. Thus, the distance between points in any 2 neighboring cells is guaranteed to be at most λ. As a result, if for a cell the total number of points in the cell and its neighbors is greater than k, then none of the points in the cell can be outliers. This property is used to eliminate the vast majority of points that cannot be outliers. Also, points belonging to cells that are more than 3 cells apart are more than a distance λ apart. As a result, if the number of points contained in all cells that are at most 3 cells away from the a given cell is less than k, then all points in the cell are definitely outliers. Finally, for those points that belong to a cell that cannot be categorized as either containing only outliers or only non-outliers, only points from neighboring cells that are at most 3 cells away need to be considered in order to determine whether or not they are outliers. Based on the above properties, the authors propose a three-pass algorithm for computing outliers in large databases. The time complexity of this cell-based algorithm is $O(c^d + N)$, where c is a number that is inversely proportional to λ. This complexity is linear with dataset size N but exponential with the number of dimensions d. As a result, due to the exponential growth in the number of cells as the number of dimensions is increased, the cell-based algorithm starts to perform poorly than the nested loop for datasets with dimensions of 4 or higher.

In [43], a similar definition of outlier is proposed. It calculates the number of points falling into the w-radius of each data point and labels those points as outliers that have low neighborhood density. We consider this definition of outliers as the same as that for $DB(k, \lambda)$-Outlier, differing only that this method does not present the threshold k explicitly in the definition. As the computation of the local density for each point is expensive, [43] proposes a clustering method for an efficient estimation. The basic idea of such approximation is to use the size of a cluster to approximate

the local density of all the data in this cluster. It uses the fix-width clustering [43] for density estimation due to its good efficiency in dealing with large data sets.

B. kNN-distance Methods

There have also been a few distance-based outlier detection methods utilizing the k nearest neighbors (kNN) in measuring the outlier-ness of data points in the dataset. The first proposal uses the distance to the k^{th} nearest neighbors of every point, denoted as D^k, to rank points so that outliers can be more efficiently discovered and ranked [101]. Based on the notion of D^k, the following definition for D_n^k-Outlier is given: Given k and n, a point is an outlier if the distance to its k^{th} nearest neighbor of the point is smaller than the corresponding value for no more than $n-1$ other points. Essentially, this definition of outliers considers the top n objects having the highest D^k values in the dataset as outliers.

Similar to the computation of $DB(k,\lambda)$-Outlier, three different algorithms, *i.e.*, the nested-loop algorithm, the index-based algorithm, and the partition-based algorithm, are proposed to compute D^k for each data point efficiently.

The *nested-loop algorithm* for computing outliers simply computes, for each input point p, D^k, the distance of between p and its k^{th} nearest neighbor. It then sorts the data and selects the top n points with the maximum D^k values. In order to compute D^k for points, the algorithm scans the database for each point p. For a point p, a list of its k nearest points is maintained, and for each point q from the database which is considered, a check is made to see if the distance between p and q is smaller than the distance of the k^{th} nearest neighbor found so far. If so, q is included in the list of the k nearest neighbors for p. The moment that the list contains more than k neighbors, then the point that is furthest away from p is deleted from the list. In this algorithm, since only one point is processed at a time, the database would need to be scanned N times, where N is the number of points in the database. The computational complexity is in the order of $O(N^2)$, which is rather expensive for large datasets. However, since we are only interested in the top n outliers, we can apply the following pruning optimization to early-stop the computation of D^k for a point p. Assume that during each step of the algorithm, we store the top n

outliers computed thus far. Let D^n_{min} be the minimum among these top n outliers. If during the computation of for a new point p, we find that the value for D^k computed so far has fallen below D^n_{min}, we are guaranteed that point p cannot be an outlier. Therefore, it can be safely discarded. This is because D^k monotonically decreases as we examine more points. Therefore, p is guaranteed not to be one of the top n outliers.

The *index-based algorithm* draws on index structure such as R*-tree [27] to speed up the computation. If we have all the points stored in a spatial index like R*-tree, the following pruning optimization can be applied to reduce the number of distance computations. Suppose that we have computed for point p by processing a portion of the input points. The value that we have is clearly an upper bound for the actual D^k of p. If the minimum distance between p and the *Minimum Bounding Rectangles* (MBR) of a node in the R*-tree exceeds the value that we have anytime in the algorithm, then we can claim that none of the points in the sub-tree rooted under the node will be among the k nearest neighbors of p. This optimization enables us to prune entire sub-trees that do not contain relevant points to the kNN search for p.

The major idea underlying the *partition-based algorithm* is to first partition the data space, and then prune partitions as soon as it can be determined that they cannot contain outliers. Partition-based algorithm is subject to the pre-processing step in which data space is split into cells and data partitions, together with the Minimum Bounding Rectangles of data partitions, are generated. Since n will typically be very small, this additional preprocessing step performed at the granularity of partitions rather than points is worthwhile as it can eliminate a significant number of points as outlier candidates. This partition-based algorithm takes the following four steps:

1. First, a clustering algorithm, such as BIRCH, is used to cluster the data and treat each cluster as a separate partition;

2. For each partition P, the lower and upper bounds (denoted as $P.lower$ and $P.upper$, respectively) on D^k for points in the partition are computed. For every point $p \in P$, we have $P.lower \leq D^k(p) \leq P.upper$;

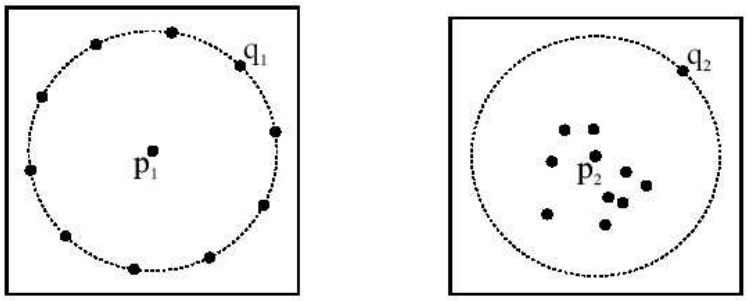

Figure 2.1: Points with the same D^k value but different outlier-ness

3. The *candidate partitions*, the partitions containing points which are candidates for outliers, are identified. Suppose we could compute $minDkDist$, the lower bound on D^k for the n outliers we have detected so far. Then, if $P.upper < minDkDist$, none of the points in P can possibly be outliers and are safely pruned. Thus, only partitions P for which $P.upper \geq minDkDist$ are chosen as candidate partitions;

4. Finally, the outliers are computed from among the points in the candidate partitions obtained in Step 3. For each candidate partition P, let $P.neighbors$ denote the neighboring partitions of P, which are all the partitions within distance $P.upper$ from P. Points belonging to neighboring partitions of P are the only points that need to be examined when computing D^k for each point in P.

The D^k_n-Outlier is further extended by considering for each point the sum of its k nearest neighbors [10]. This extension is motivated by the fact that the definition of D^k merely considers the distance between an object with its k^{th} nearest neighbor, entirely ignoring the distances between this object and its another $k-1$ nearest neighbors. This drawback may make D^k fail to give an accurate measurement of outlier-ness of data points in some cases. For a better understanding, we present an example, as shown in Figure 2.1, in which the same D^k value is assigned to points p_1 and p_2, two points with apparently rather different outlier-ness. The $k-1$ nearest neighbors for p_2 are populated much more densely around it than those of p_1, thus the outlier-ness of p_2 is obviously lower than p_1. Obviously, D^k is not robust enough

in this example to accurately reveal the outlier-ness of data points. By summing up the distances between the object with all of its k nearest neighbors, we will be able to have a more accurate measurement of outlier-ness of the object, though this will require more computational effort in summing up the distances. This method is also used in [43] for anomaly detection.

The idea of kNN-based distance metric can be extended to consider the k nearest dense regions. The recent methods are the Largest_cluster method [79][125] and Grid-ODF [114], as discussed below.

Khoshgoftaar *et al.* propose a distance-based method for labeling wireless network traffic records in the data stream used as either normal or intrusive [79][125]. Let d be the largest distance of an instance to the centriod of the largest cluster. Any instance or cluster that has a distance greater than αd ($\alpha \geq 1$) to the largest cluster is defined as an attack. This method is referred to as the Largest_Cluster method. It can also be used to detect outliers. It takes the following several steps for outlier detection:

1. Find the largest cluster, *i.e.* the cluster with largest number of instances, and label it as normal. Let c_0 be the centriod of this cluster;

2. Sort the remaining clusters in ascending order based on the distance from their cluster centroid to c_0;

3. Label all the instances that have a distance to c_0 greater than αd, where α is a human-specified parameter;

4. Label all the other instances as normal.

When used in dealing with projected anomalies detection for high-dimensional data streams, this method suffers the following limitations:

- First and most importantly, this method does not take into account the nature of outliers in high-dimensional data sets and is unable to explore subspaces to detect projected outliers;

- k-means clustering is used in this method as the backbone enabling technique for detecting intrusions. This poses difficulty for this method to deal with data streams. k-means clustering requires iterative optimization of clustering centroids to gradually achieve better clustering results. This optimization process involves multiple data scans, which is infeasible in the context of data streams;

- A strong assumption is made in this method that all the normal data will appear in a single cluster (*i.e.*, the largest cluster), which is not properly substantiated in the paper. This assumption may be too rigid in some applications. It is possible that the normal data are distributed in two or more clusters that correspond to a few varying normal behaviors. For a simple instance, the network traffic volume is usually high during the daytime and becomes low late in the night. Thus, network traffic volume may display several clusters to represent behaviors exhibiting at different time of the day. In such case, the largest cluster is apparently not where all the normal cases are only residing;

- In this method, one needs to specify the parameter α. The method is rather sensitive to this parameter whose best value is not obvious whatsoever. First, the distance scale between data will be rather different in various subspaces; the distance between any pair of data is naturally increased when it is evaluated in a subspace with higher dimension, compared to in a lower-dimensional subspace. Therefore, specifying an ad-hoc α value for each subspace evaluated is rather tedious and difficult. Second, α is also heavily affected by the number of clusters the clustering method produces, *i.e.*, k. Intuitively, when the number of clusters k is small, D will become relatively large, then α should be set relatively small accordingly, and vice versa.

Recently, an extension of the notion of kNN, called Grid-ODF, from the k nearest objects to the k nearest dense regions is proposed [114]. This method employed the sum of the distances between each data point and its k nearest dense regions to rank data points. This enables the algorithm to measure the outlier-ness of data points from a more global perspective. Grid-ODF takes into account the mechanisms used

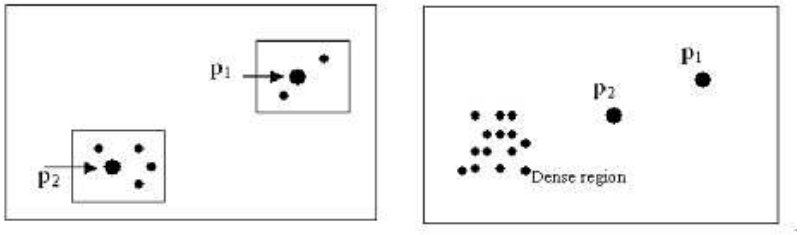

Figure 2.2: Local and global perspectives of outlier-ness of p_1 and p_2

in detecting both global and local outliers. In the local perspective, human examine the point's immediate neighborhood and consider it as an outlier if its neighborhood density is low. The global observation considers the dense regions where the data points are densely populated in the data space. Specifically, the neighboring density of the point serves as a good indicator of its outlying degree from the local perspective. In the left sub-figure of Figure 2.2, two square boxes of equal size are used to delimit the neighborhood of points p_1 and p_2. Because the neighboring density of p_1 is less than that of p_2, so the outlying degree of p_1 is larger than p_2. On the other hand, the distance between the point and the dense regions reflects the similarity between this point and the dense regions. Intuitively, the larger such distance is, the more remarkably p is deviated from the main population of the data points and therefore the higher outlying degree it has, otherwise it is not. In the right sub-figure of 2.2, we can see a dense region and two outlying points, p_1 and p_2. Because the distance between p_1 and the dense region is larger than that between p_2 and the dense region, so the outlying degree of p_1 is larger than p_2.

Based on the above observations, a new measurement of outlying factor of data points, called *Outlying Degree Factor* (ODF), is proposed to measure the outlier-ness of points from both the global and local perspectives. The ODF of a point p is defined as follows:

$$ODF(p) = \frac{k\_DF(p)}{NDF(p)}$$

where $k\_DF(p)$ denotes the average distance between p and its k nearest dense cells and $NDF(p)$ denotes number of points falling into the cell to which p belongs.

In order to implement the computation of ODF of points efficiently, grid structure

is used to partition the data space. The main idea of grid-based data space partition is to super-impose a multi-dimensional cube in the data space, with equal-volumed cells. It is characterized by the following advantages. First, $NDF(p)$ can be obtained instantly by simply counting the number of points falling into the cell to which p belongs, without the involvement of any indexing techniques. Secondly, the dense regions can be efficiently identified, thus the computation of $k\_DF(p)$ can be very fast. Finally, based on the density of grid cells, we will be able to select the top n outliers only from a specified number of points viewed as *outlier candidates*, rather than the whole dataset, and the final top n outliers are selected from these outlier candidates based on the ranking of their ODF values.

The number of outlier candidates is typically 9 or 10 times as large as the number of final outliers to be found (*i.e.*, top n) in order to provide a sufficiently large pool for outlier selection. Let us suppose that the size of outlier candidates is $m * n$, where the m is a positive number provided by users. To generate $m * n$ outlier candidates, all the cells containing points are sorted in ascending order based on their densities, and then the points in the first t cells in the sorting list that satisfy the following inequality are selected as the $m * n$ outlier candidates:

$$\sum_{i=1}^{t-1} Den(C_i) \leq m * n \leq \sum_{i=1}^{t} Den(C_i)$$

The kNN-distance methods, which define the top n objects having the highest values of the corresponding outlier-ness metrics as outliers, are advantageous over the local neighborhood methods in that they order the data points based on their relative ranking, rather than on the distance cutoff. Since the value of n, the top outlier users are interested in, can be very small and is relatively independent of the underlying data set, it will be easier for the users to specify compared to the distance threshold λ.

C. Advantages and Disadvantages of Distance-based Methods

The major advantage of distance-based algorithms is that, unlike distribution-based methods, distance-based methods are non-parametric and do not rely on any

assumed distribution to fit the data. The distance-based definitions of outliers are fairly straightforward and easy to understand and implement.

Their major drawback is that most of them are not effective in high-dimensional space due to the curse of dimensionality, though one is able to mechanically extend the distance metric, such as Euclidean distance, for high-dimensional data. The high-dimensional data in real applications are very noisy, and the abnormal deviations may be embedded in some lower-dimensional subspaces that cannot be observed in the full data space. Their definitions of a local neighborhood, irrespective of the circular neighborhood or the k nearest neighbors, do not make much sense in high-dimensional space. Since each point tends to be equi-distant with each other as number of dimensions goes up, the degree of outlier-ness of each points are approximately identical and significant phenomenon of deviation or abnormality cannot be observed. Thus, none of the data points can be viewed outliers if the concepts of proximity are used to define outliers. In addition, neighborhood and kNN search in high-dimensional space is a non-trivial and expensive task. Straightforward algorithms, such as those based on nested loops, typically require $O(N^2)$ distance computations. This quadratic scaling means that it will be very difficult to mine outliers as we tackle increasingly larger data sets. This is a major problem for many real databases where there are often millions of records. Thus, these approaches lack a good scalability for large data set. Finally, the existing distance-based methods are not able to deal with data streams due to the difficulty in maintaining a data distribution in the local neighborhood or finding the kNN for the data in the stream.

2.2.3 Density-based Methods

Density-based methods use more complex mechanisms to model the outlier-ness of data points than distance-based methods. It usually involves investigating not only the local density of the point being studied but also the local densities of its nearest neighbors. Thus, the outlier-ness metric of a data point is relative in the sense that it is normally a ratio of density of this point against the the averaged densities of its nearest neighbors. Density-based methods feature a stronger modeling capability

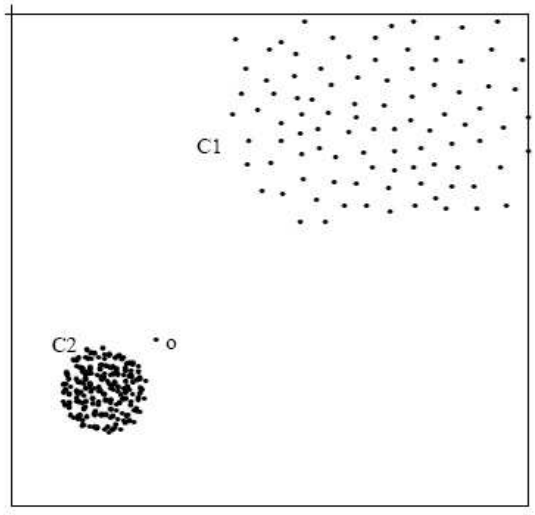

Figure 2.3: A sample dataset showing the advantage of LOF over $DB(k, \lambda)$-Outlier

of outliers but require more expensive computation at the same time. What will be discussed in this subsection are the major density-based methods called LOF method, COF method, INFLO method and MDEF method.

A. LOF Method

The first major density-based formulation scheme of outlier has been proposed in [26], which is more robust than the distance-based outlier detection methods. An example is given in [26] (refer to figure 2.3), showing the advantage of a density-based method over the distance-based methods such as $DB(k, \lambda)$-Outlier. The dataset contains an outlier o, and C_1 and C_2 are two clusters with very different densities. The $DB(k, \lambda)$-Outlier method cannot distinguish o from the rest of the data set no matter what values the parameters k and λ take. This is because the density of o's neighborhood is very much closer to the that of the points in cluster C_1. However, the density-based method, proposed in [26], can handle it successfully.

This density-based formulation quantifies the outlying degree of points using *Local Outlier Factor (LOF)*. Given parameter $MinPts$, LOF of a point p is defined as

$$LOF_{MinPts}(p) = \frac{\sum_{o \in MinPts(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

where $|N_{MinPts}(p)|$ denotes the number of objects falling into the $MinPts$-neighborhood

of p and $lrd_{MinPts}(p)$ denotes the *local reachability density* of point p that is defined as the inverse of the average reachability distance based on the $MinPts$ nearest neighbors of p, *i.e.*,

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o \in MinPts(p)} reach\_dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right)$$

Further, the reachability distance of point p is defined as

$$reach\_dist_{MinPts}(p,o) = max(MinPts\_distance(o), dist(p,o))$$

Intuitively speaking, LOF of an object reflects the density contrast between its density and those of its neighborhood. The neighborhood is defined by the distance to the $MinPts^{th}$ nearest neighbor. The local outlier factor is a mean value of the ratio of the density distribution estimate in the neighborhood of the object analyzed to the distribution densities of its neighbors [26]. The lower the density of p and/or the higher the densities of p's neighbors, the larger the value of $LOF(p)$, which indicates that p has a higher degree of being an outlier. A similar outlier-ness metric to LOF, called OPTICS-OF, was proposed in [25].

Unfortunately, the LOF method requires the computation of LOF for all objects in the data set which is rather expensive because it requires a large number of kNN search. The high cost of computing LOF for each data point p is caused by two factors. First, we have to find the $MinPts^{th}$ nearest neighbor of p in order to specify its neighborhood. This resembles to computing D^k in detecting D^k_n-Outliers. Secondly, after the $MinPts^{th}$-neighborhood of p has been determined, we have to further find the $MinPts^{th}$-neighborhood for each data points falling into the $MinPts^{th}$-neighborhood of p. This amounts to $MinPts^{th}$ times in terms of computation efforts as computing D^k when we are detecting D^k_n-Outliers.

It is desired to constrain a search to only the top n outliers instead of computing the LOF of every object in the database. The efficiency of this algorithm is boosted by an efficient micro-cluster-based local outlier mining algorithm proposed in [66].

LOF ranks points by only considering the neighborhood density of the points, thus it may miss out the potential outliers whose densities are close to those of their neighbors. Furthermore, the effectiveness of this algorithm using LOF is rather sensitive

to the choice of $MinPts$, the parameter used to specify the local neighborhood.

B. COF Method

As LOF method suffers the drawback that it may miss those potential outliers whose local neighborhood density is very close to that of its neighbors. To address this problem, Tang *et al.* proposed a new *Connectivity-based Outlier Factor* (COF) scheme that improves the effectiveness of LOF scheme when a pattern itself has similar neighborhood density as an outlier [111]. In order to model the connectivity of a data point with respect to a group of its neighbors, a set-based nearest path (SBN-path) and further a set-based nearest trail (SBN-trail), originated from this data point, are defined. This SNB trail stating from a point is considered to be the pattern presented by the neighbors of this point. Based on SNB trail, the cost of this trail, a weighted sum of the cost of all its constituting edges, is computed. The final outlier-ness metric, COF, of a point p with respect to its k-neighborhood is defined as

$$COF_k(p) = \frac{|N_k(p)| * ac\_dist_{N_k(p)}(p)}{\sum_{o \in N_k(p)} ac\_dist_{N_k(o)}(o)}$$

where $ac\_dist_{N_k(p)}(p)$ is the average chaining distance from point p to the rest of its k nearest neighbors, which is the weighted sum of the cost of the SBN-trail starting from p.

It has been shown in [111] that COF method is able to detect outlier more effectively than LOF method for some cases. However, COF method requires more expensive computations than LOF and the time complexity is in the order of $O(N^2)$ for high-dimensional datasets.

C. INFLO Method

Even though LOF is able to accurately estimate outlier-ness of data points in most cases, it fails to do so in some complicated situations. For instance, when outliers are in the location where the density distributions in the neighborhood are significantly different, this may result in a wrong estimation. An example where LOF fails to have an accurate outlier-ness estimation for data points has been given in [67]. The example is presented in Figure 2.4. In this example, data p is in fact part of a sparse cluster $C2$ which is near the dense cluster $C1$. Compared to objects q and r, p

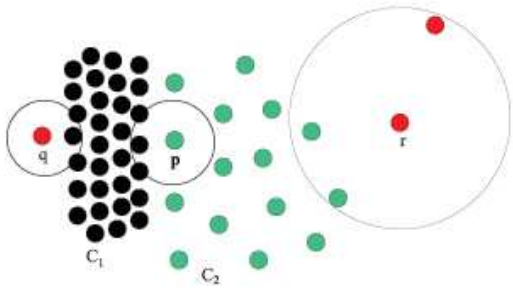

Figure 2.4: An example where LOF does not work

obviously displays less outlier-ness. However, if LOF is used in this case, p could be mistakenly regarded to having stronger outlier-ness than q and r.

Authors in [67] pointed out that this problem of LOF is due to the inaccurate specification of the space where LOF is applied. To solve this problem of LOF, an improved method, called INFLO, is proposed [67]. The idea of INFLO is that both the nearest neighbors (NNs) and reverse nearest neighbors (RNNs) of a data point are taken into account in order to get a better estimation of the neighborhood's density distribution. The RNNs of an object p are those data points that have p as one of their k nearest neighbors. By considering the symmetric neighborhood relationship of both NN and RNN, the space of an object influenced by other objects is well determined. This space is called the *k-influence space* of a data point. The outlier-ness of a data point, called INFLuenced Outlierness (INFLO), is quantified. INFLO of a data point p is defined as

$$INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)}$$

INFLO is by nature very similar to LOF. With respect to a data point p, they are both defined as the ratio of p's its density and the average density of its neighboring objects. However, INFLO uses only the data points in its k-influence space for calculating the density ratio. Using INFLO, the densities of its neighborhood will be reasonably estimated, and thus the outliers found will be more meaningful.

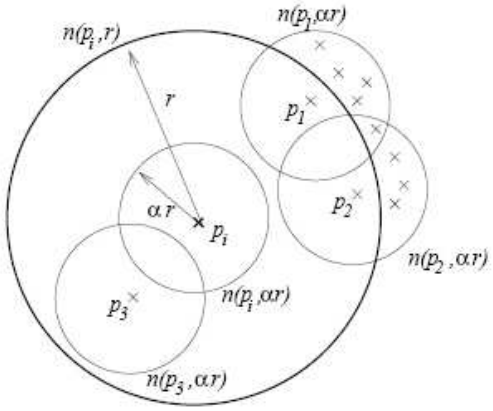

Figure 2.5: Definition of MDEF

D. MDEF Method

In [95], a new density-based outlier definition, called Multi-granularity Deviation Factor (MEDF), is proposed. Intuitively, the MDEF at radius r for a point p_i is the relative deviation of its local neighborhood density from the average local neighborhood density in its r-neighborhood. Let $n(p_i, \alpha r)$ be the number of objects in the αr-neighborhood of p_i and $\hat{n}(p_i, r, \alpha)$ be the average, over all objects p in the r-neighborhood of p_i, of $n(p, \alpha r)$. In the example given by Figure 2.5, we have $n(p_i, \alpha r) = 1$, and $\hat{n}(p_i, r, \alpha) = (1 + 6 + 5 + 1)/4 = 3.25$.

MDEF of p_i, given r and α, is defined as

$$MDEF(p_i, r, \alpha) = 1 - \frac{n(p_i, \alpha r)}{\hat{n}(pi, r, \alpha)}$$

where $\alpha = \frac{1}{2}$. A number of different values are set for the sampling radius r and the minimum and the maximum values for r are denoted by r_{min} and r_{max}. A point is flagged as an outliers if for any $r \in [r_{min}, r_{max}]$, its MDEF is sufficient large.

E. Advantages and Disadvantages of Density-based Methods

The density-based outlier detection methods are generally more effective than the distance-based methods. However, in order to achieve the improved effectiveness, the density-based methods are more complicated and computationally expensive. For a data object, they have to not only explore its local density but also that of its

neighbors. Expensive kNN search is expected for all the existing methods in this category. Due to the inherent complexity and non-updatability of their outlier-ness measurements used, LOF, COF, INFLO and MDEF cannot handle data streams efficiently.

2.2.4 Clustering-based Methods

The final category of outlier detection algorithm for relatively low dimensional static data is clustering-based. Many data-mining algorithms in literature find outliers as a by-product of clustering algorithms [6, 11, 13, 55, 129] themselves and define outliers as points that do not lie in or located far apart from any clusters. Thus, the clustering techniques implicitly define outliers as the background noise of clusters. So far, there are numerous studies on clustering, and some of them are equipped with some mechanisms to reduce the adverse effect of outliers, such as CLARANS [91], DBSCAN [44], BIRCH [129], WaveCluster [104]. More recently, we have seen quite a few clustering techniques tailored towards subspace clustering for high-dimensional data including CLIQUE [6] and HPStream [9].

Next, we will review several major categories of clustering methods, together with the analysis on their advantages and disadvantages and their applicability in dealing with outlier detection problem for high-dimensional data streams.

A. Partitioning Clustering Methods

The partitioning clustering methods perform clustering by partitioning the data set into a specific number of clusters. The number of clusters to be obtained, denoted by k, is specified by human users. They typically start with an initial partition of the dataset and then iteratively optimize the objective function until it reaches the optimal for the dataset. In the clustering process, center of the clusters (centroid-based methods) or the point which is located nearest to the cluster center (medoid-based methods) is used to represent a cluster. The representative partitioning clustering methods are PAM, CLARA, k-means and CLARANS.

PAM [80] uses a k-medoid method to identify the clusters. PAM selects k objects arbitrarily as medoids and swap with objects until all k objects qualify as medoids.

PAM compares an object with entire dataset to find a medoid, thus it has a slow processing time with a complexity of $\mathcal{O}(k(N-k)^2)$, where N is number of data in the data set and k is the number of clusters.

CLARA [80] tries to improve the efficiency of PAM. It draws a sample from the dataset and applies PAM on the sample that is much smaller in size than the the whole dataset.

k-means [86] initially choose k data objects as seeds from the dataset. They can be chosen randomly or in a way such that the points are mutually farthest apart. Then, it examines each point in the dataset and assigns it to one of the clusters depending on the minimum distance. The centroid's position is recalculated and updated the moment a point is added to the cluster and this continues until all the points are grouped into the final clusters. The k-means algorithm is relatively scalable and efficient in processing large datasets because the computational complexity is $\mathcal{O}(nkt)$, where n is total number of points, k is the number of clusters and t is the number of iterations of clustering. However, because it uses a centroid to represent each cluster, k-means suffers the inability to correctly cluster with a large variation of size and arbitrary shapes, and it is also very sensitive to the noise and outliers of the dataset since a small number of such data will substantially effect the computation of mean value the moment a new object is clustered.

CLARANS [91] is an improved k-medoid method, which is based on randomized search. It begins with a random selection of k nodes, and in each of following steps, compares each node to a specific number of its neighbors in order to find a local minimum. When one local minimum is found, CLARANS continues to repeat this process for another minimum until a specific number of minima have been found. CLARANS has been experimentally shown to be more effective than both PAM and CLEAR. However, the computational complexity of CLARANS is close to quadratic w.r.t the number of points [113], and it is prohibitive for clustering large database. Furthermore, the quality of clustering result is dependent on the sampling method, and it is not stable and unique due to the characteristics of randomized search.

B. Hierarchical Clustering Methods

Hierarchical clustering methods essentially constructs a hierarchical decomposition of the whole dataset. It can be further divided into two categories based on how this dendrogram is operated to generate clusters, *i.e.*, *agglomerative methods* and *divisive methods*. An agglomerative method begins with each point as a distinct cluster and merges two closest clusters in each subsequent step until a stopping criterion is met. A divisive method, contrary to an agglomerative method, begins with all the point as a single cluster and splits it in each subsequent step until a stopping criterion is met. Agglomerative methods are seen more popular in practice. The representatives of hierarchical methods are MST clustering, CURE and CHAMELEON.

MST clustering [118] is a graph-based divisive clustering algorithm. Given n points, a MST is a set of edges that connects all the points and has a minimum total length. Deletion of edges with larger lengths will subsequently generate a specific number of clusters. The overhead for MST clustering is determined by the Euclidean MST construction, which is $\mathcal{O}(nlogn)$ in time complexity, thus MST algorithm can be used for scalable clustering. However, MST algorithm can only work well on the clean dataset and are sensitive to outliers. The intervention of outliers, termed "chaining-effect" (that is, a line of outliers between two distinct clusters will make these two clusters be marked as one cluster due to its adverse effect), will seriously degrade the quality of the clustering results.

CURE [55] employs a novel hierarchical clustering algorithm in which each cluster is represented by a constant number of well-distributed points. A random sample drawn from the original dataset is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters. The multiple representative points for each cluster are picked to be as disperse as possible and shrink towards the center using a pre-specified shrinking factor. At each step of the algorithm, the two clusters with the closest pair of representative (this pair of representative points are from different clusters) points are merged. Usage of multiple points representing a cluster enables CURE to well capture the shape of clusters and makes it suitable for clusters with non-spherical

shapes and wide variance in size. The shrinking factor helps to dampen the adverse effect of outliers. Thus, CURE is more robust to outliers and identifies clusters having arbitrary shapes.

CHAMELEON [72] is a clustering technique trying to overcome the limitation of existing agglomerative hierarchical clustering algorithms that the clustering is irreversible. It operates on a sparse graph in which nodes represent data points and weighted edges represent similarities of among the data points. CHAMELEON first uses a graph partition algorithm to cluster the data points into a large number of relatively small sub-clusters. It then employs an agglomerative hierarchical clustering algorithm to genuine clusters by progressively merging these sub-clusters. The key feature of CHAMELEON lies in its mechanism determining the similarity between two sub-clusters in sub-cluster merging. Its hierarchical algorithm takes into consideration of both inter-connectivity and closeness of clusters. Therefore, CHAMELEON can dynamically adapt to the internal characteristics of the clusters being merged.

C. Density-based Clustering Methods

The density-based clustering algorithms consider normal clusters as dense regions of objects in the data space that are separated by regions of low density. Human normally identify a cluster because there is a relatively denser region compared to its sparse neighborhood. The representative density-based clustering algorithms are DBSCAN and DENCLUE.

The key idea of DBSCAN [44] is that for each point in a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN introduces the notion of "density-reachable points" and based on which performs clustering. In DBSCAN, a cluster is a maximum set of density-reachable points w.r.t. parameters Eps and $MinPts$, where Eps is the given radius and $MinPts$ is the minimum number of points required to be in the Eps-neighborhood. Specifically, to discover clusters in the dataset, DBSCAN examines the Eps-neighborhood of each point in the dataset. If the Eps-neighborhood of a point p contains more than $MinPts$, a new cluster with p as the core object is generated. All the objects from within this Eps-neighborhood are then assigned to this cluster. All this newly entry points

will also go through the same process to gradually grow this cluster. When there is no more core object can be found, another core object will be initiated and another cluster will grow. The whole clustering process terminates when there are no new points can be added to any clusters. As the clusters discovered are dependent on the specification of the parameters, DBSCAN relies on the user's ability to select a good set of parameters. DBSCAN outperforms CLARANS by a factor of more than 100 in terms of efficiency [44]. DBSCAN is also powerful in discovering of clusters with arbitrary shapes. The drawbacks DBSCAN suffers are: (1) It is subject to adverse effect resulting from "chaining-effect"; (2) The two parameters used in DBSCAN, *i.e.*, Eps and $MinPts$, cannot be easily decided in advance and require a tedious process of parameter tuning.

DENCLUE [61] performs clustering based on density distribution functions, a set of mathematical functions used to model the influence of each point within its neighborhood. The overall density of the data space can be modeled as sum of influence function of all data points. The clusters can be determined by density attractors. Density attractors are the local maximum of the overall density function. DENCLUE has advantages that it can well deal with dataset with a large number of noises and it allows a compact description of clusters of arbitrary shape in high-dimensional datasets. To facilitate the computation of the density function, DENCLUE makes use of grid-like structure. Noted that even though it uses grids in clustering, DENCLUE is fundamentally different from grid-based clustering algorithm in that grid-based clustering algorithm uses grid for summarizing information about the data points in each grid cell, while DENCLUE uses such structure to effectively compute the sum of influence functions at each data point.

D. Grid-based Clustering Methods

Grid-based clustering methods perform clustering based on a grid-like data structure with the aim of enhancing the efficiency of clustering. It quantizes the space into a finite number of cells which form a grid structure on which all the operations for clustering are performed. The main advantage of the approaches in this category is their fast processing time which is typically only dependent on the number of cells in

the quantized space, rather than the number of data objects. The representatives of grid-based clustering algorithms are STING, WaveCluster and DClust.

STING [113] divides the spatial area into rectangular grids, and builds a hierarchical rectangle grids structure. It scans the dataset and computes the necessary statistical information, such as mean, variance, minimum, maximum, and type of distribution, of each grid. The hierarchical grid structure can represent the statistical information with different resolutions at different levels. The statistical information in this hierarchical structure can be used to answer queries. The likelihood that a cell is relevant to the query at some confidence level is computed using the parameters of this cell. The likelihood can be defined as the proportion of objects in this cell that satisfy the query condition. After the confidence interval is obtained, the cells are labeled as relevant or irrelevant based on some confidence threshold. After examining the current layer, the clustering proceeds to the next layer and repeats the process. The algorithm will subsequently only examine the relevant cells instead of all the cells. This process terminates when all the layers have been examined. In this way, all the relevant regions (clusters) in terms of query are found and returned.

WaveCluster [104] is grid-based clustering algorithm based on wavelet transformation, a commonly used technique in signal processing. It transforms the multi-dimensional spatial data to the multi-dimensional signal, and it is able to identify dense regions in the transformed domain that are clusters to be found.

In DClust [122], the data space involved is partitioned into cells with equal size and data points are mapped into the grid structure. A number of representative points of the database are picked using the density criterion. A *Minimum Spanning Tree* (MST) of these representative points, denoted as R-MST, is built. After the R-MST has been constructed, multi-resolution clustering can be easily achieved. Suppose a user wants to find k clusters. A graph search through the R-MST is initiated, starting from the largest cost edge, to the lowest cost edge. As an edge is traversed, it is marked as deleted from the R-MST. The number of partitions resulting from the deletion is computed. The process stops when the number of partitions reaches k. Any change in the value of k simply implies re-initiating the search-and-marked

procedure on the R-MST. Once the R-MST has been divided into k partitions, we can now propagate this information to the original dataset so that each point in the dataset is assigned to one and only one partition/cluster. DClust is equipped with more robust outlier elimination mechanisms to identify and filter the outliers during the various stages of the clustering process. First, DClust uses a uniform random sampling approach to sample the large database. This is effective in ruling out the majority of outliers in the database. Hence, the sample database obtained will be reasonably clean; Second, DClust employs a grid structure to identify representative points. Grid cells whose density is less than the threshold are pruned. This pre-filtering step ensures that the R-MST constructed is an accurate reflection of the underlying cluster structure. Third, the clustering of representative points may cause a number of the outliers that are in close vicinity to form a cluster. The number of points in such outlier clusters will be much smaller than the number of points in the normal clusters. Thus, any small clusters of representative points will be treated as outlier clusters and eliminated. Finally, when the points in the dataset are labeled, some of these points may be quite far from any representative point. DClust will regard such points as outliers and filter them out in the final clustering results.

E. Advantages and Disadvantage of Clustering-based Methods

Detecting outliers by means of clustering analysis is quite intuitive and consistent with human perception of outliers. In addition, clustering is a well-established research area and there have been abundant clustering algorithms that users can choose from for performing clustering and then detecting outliers.

Nevertheless, many researchers argue that, strictly speaking, clustering algorithms should not be considered as outlier detection methods, because their objective is only to group the objects in dataset such that clustering functions can be optimized. The aim to eliminate outliers in dataset using clustering is only to dampen their adverse effect on the final clustering result. This is in contrast to the various definitions of outliers in outlier detection which are more objective and independent of how clusters in the input data set are identified. One of the major philosophies in designing new

outlier detection approaches is to directly model outliers and detect them without going though clustering the data first. In addition, the notions of outliers in the context of clustering are essentially binary in nature, without any quantitative indication as to how outlying each object is. It is desired in many applications that the outlier-ness of the outliers can be quantified and ranked.

2.3 Outlier Detection Methods for High Dimensional Data

There are many applications in high-dimensional domains in which the data can contain dozens or even hundreds of dimensions. The outlier detection techniques we have reviewed in the preceding sections use various concepts of proximity in order to find the outliers based on their relationship to the other points in the data set. However, in high-dimensional space, the data are sparse and concepts using the notion of proximity fail to achieve most of their effectiveness. This is due to the curse of dimensionality that renders the high-dimensional data tend to be equi-distant to each other as dimensionality increases. They does not consider the outliers embedded in subspaces and are not equipped with the mechanism for detecting them.

2.3.1 Methods for Detecting Outliers in High-dimensional Data

To address the challenge associated with high data dimensionality, two major categories of research work have been conducted. The first category of methods project the high dimensional data to lower dimensional data. Dimensionality deduction techniques, such as *Principal Component Analysis*(PCA), *Independent Component Analysis* (ICA), *Singular Value Decomposition* (SVD), etc can be applied to the high-dimensional data before outlier detection is performed. Essentially, this category of methods perform feature selection and can be considered as the pre-processing work for outlier detection. The second category of approaches is more promising yet challenging. They try to re-design the mechanism to accurately capture the proximity relationship between data points in the high-dimensional space [14].

A. Sparse Cube Method

Aggarwal *et al.* conducted some pioneering work in high-dimensional outlier detection [15][14]. They proposed a new technique for outlier detection that finds outliers by observing the density distributions of projections from the data. This new definition considers a point to be an outlier if in some lower-dimensional projection it is located in a local region of abnormally low density. Therefore, the outliers in these lower-dimensional projections are detected by simply searching for these projections featuring lower density. To measure the sparsity of a lower-dimensional projection quantitatively, the authors proposed the so-called *Sparsity Coefficient*. The computation of Sparsity Coefficient involves a grid discretization of the data space and making an assumption of normal distribution for the data in each cell of the hypercube. Each attribute of the data is divided into φ equi-depth ranges. In each range, there is a fraction $f = 1/\varphi$ of the data. Then, a k-dimensional cube is made of ranges from k different dimensions. Let N be the dataset size and $n(D)$ denote the number of objects in a k-dimensional cube D. Under the condition that attributes were statistically independent, the Sparsity Coefficient $S(D)$ of the cube D is defined as:

$$S(D) = \frac{n(D) - N * f^k}{\sqrt{N * f^k * (1 - f^k)}}$$

Since there are no closure properties for Sparsity Coefficient, thus no fast subspace pruning can be performed and the lower-dimensional projection search problem becomes a NP-hard problem. Therefore, the authors employ evolutionary algorithm in order to solve this problem efficiently. After lower-dimensional projections have been found, a post-processing phase is required to map these projections into the data points; all the sets of data points that contain in the abnormal projections reported by the algorithm.

B. Example-based Method

Recently, an approach using outlier examples provided by users are used to detect outliers in high-dimensional space [123][124]. It adopts an *"outlier examples* $\rightarrow$

subspaces $\rightarrow$ *outliers*$''$ manner to detect outliers. Specifically, human users or domain experts first provide the systems with a few initial outlier examples. The algorithm finds the subspaces in which most of these outlier examples exhibit significant outlier-ness. Finally, other outliers are detected from these subspaces obtained in the previous step. This approach partitions the data space into equi-depth cells and employs the Sparsity Coefficient proposed in [14] to measure the outlier-ness of outlier examples in each subspace of the lattice. Since it is untenable to exhaustively search the space lattice, the author also proposed to use evolutionary algorithms for subspace search. The fitness of a subspace is the average Sparsity Coefficients of all cubes in that subspace to which the outlier examples belong. All the objects contained in the cubes which are sparser than or as sparse as cubes containing outlier examples in the subspace are detected as outliers.

However, this method is limited in that it is only able to find the outliers in the subspaces where most of the given user examples are outlying significantly. It cannot detect those outliers that are embedded in other subspaces. Its capability for effective outlier detection is largely depended on the number of given examples and, more importantly, how these given examples are similar to the majority of outliers in the dataset. Ideally, this set of user examples should be a good sample of all the outliers in the dataset. This method works poorly when the number of user examples is quite small and cannot provide enough clues as to where the majority of outliers in the dataset are. Providing such a good set of outlier examples is a difficult task whatsoever. The reasons are two-fold. First, it is not trivial to obtain a set of outlier examples for a high-dimensional data set. Due to a lack of visualization aid in high-dimensional data space, it is not obvious at all to find the initial outlier examples unless they are detected by some other techniques. Secondly and more importantly, even when a set of outliers have already been obtained, testing the representativeness of this outlier set is almost impossible. Given these two strong constraints, this approach becomes inadequate in detecting outliers in high-dimensional datasets. It will miss out those projected outliers that are not similar to those given outlier examples.

C. Outlier Detection in Subspaces

Since outlier-ness of data points mainly appear significant in some subspaces of moderate dimensionality in high-dimensional space and the quality of the outliers detected varies in different subspaces consisting of different combinations of dimension subsets. The authors in [29] employ evolutionary algorithm for feature selection (find optimal dimension subsets which represent the original dataset without losing information for unsupervised learning task of outlier detection as well as clustering). This approach is a wrapper algorithm in which the dimension subsets are selected such that the quality of outlier detected or the clusters generated can be optimized. The originality of this work is to combine the evolutionary algorithm with the data visualization technique utilizing parallel coordinates to present evolution results interactively and allow users to actively participate in evolutionary algorithm searching to achieve a fast convergence of the algorithm.

D. Subspace Outlier Detection for Categorical Data

Das *et al.* study the problem of detecting anomalous records in categorical data sets [40]. They draw on a probability approach for outlier detection. For each record in the data set, the probabilities for the occurrence of different subsets of attributes are investigated. A data record is labeled as an outlier if the occurrence probability for the values of some of its attribute subsets is quite low. Specifically, the probability for two subsets of attributes a_t and b_t to occur together in a record, denoted by $r(a_t, b_t)$, is quantified as:

$$r(a_t, b_t) = \frac{P(a_t, b_t)}{P(a_t)P(b_t)}$$

Due to the extremely large number of possible attribute subsets, only the attribute subsets with a length not exceeding than k are studied.

Because it always evaluates pairs of attribute subsets, each of which contain at least one attribute, therefore, this method will miss out the abnormality evaluation for 1-dimensional attribute subsets. In addition, due to the exponential growth of the number of attribute subsets w.r.t k, the value of k is set typically small in this method. Hence, this method can only cover attribute subsets not larger than $2k$ for

a record (this method evaluates a pair of attribute subsets at a time). This limits the ability of this method for detecting records that have outlying attribute subsets larger than $2k$.

2.3.2 Outlying Subspace Detection for High-dimensional Data

All the outlier detection algorithms that we have discussed so far, regardless of in low or high dimensional scenario, invariably fall into the framework of detecting outliers in a specific data space, either in full space or subspace. We term these methods as "$space \rightarrow outliers$" techniques. For instance, outliers are detected by first finding locally sparse subspaces [14], and the so-called Strongest/Weak Outliers are discovered by first finding the Strongest Outlying Spaces [77].

A new research problem called *outlying subspace detection* for multiple or high dimensional data has been identified recently in [121][126][120]. The major task of outlying subspace detection is to find those subspaces (subset of features) in which the data points of interest exhibit significant deviation from the rest of population. This problem can be formulated as follows: given a data point or object, find the subspaces in which this data is considerably dissimilar, exceptional or inconsistent with respect to the remaining points or objects. These points under study are called *query points*, which are usually the data that users are interested in or concerned with. As in [121][126], a distance threshold T is utilized to decide whether or not a data point deviates significantly from its neighboring points. A subspace s is called an outlying subspace of data point p if $OD_s(p) \geq T$, where OD is the outlier-ness measurement of p.

Finding the correct subspaces so that outliers can be detected is informative and useful in many practical applications. For example, in the case of designing a training program for an athlete, it is critical to identify the specific subspace(s) in which an athlete deviates from his or her teammates in the daily training performances. Knowing the specific weakness (subspace) allows a more targeted training program to be designed. In a medical system, it is useful for the Doctors to identify from voluminous medical data the subspaces in which a particular patient is found abnormal and

therefore a corresponding medical treatment can be provided in a timely manner.

The unique feature of the problem of outlying subspace detection is that, instead of detecting outliers in specific subspaces as did in the classical outlier detection techniques, it involves searching from the space lattice for the associated subspaces whereby the given data points exhibit abnormal deviations. Therefore, the problem of outlying subspace detection is called an $''outlier \rightarrow spaces''$ problem so as to distinguish the classical outlier detection problem which is labeled as a $''space \rightarrow outliers''$ problem. It has been theoretically and experimentally shown that the conventional outlier detection methods, irrespectively dealing with low or high-dimensional data, cannot successfully cope with the problem of outlying subspace detection problem in [121]. The existing high-dimensional outlier detection techniques, *i.e.*, find outliers in given subspaces, are theoretically applicable to solve the outlying detection problem. To do this, we have to detect outliers in all subspaces and a search in all these subspaces is needed to find the set of outlying subspaces of p, which are those subspaces in which p is in their respective set of outliers. Obviously, the computational and space costs are both in an exponential order of d, where d is the number of dimensions of the data point. Such an exhaustive space searching is rather expensive in high-dimensional scenario. In addition, they usually only return the top n outliers in a given subspace, thus it is impossible to check whether or not p is an outlier in this subspace if p is not in this top n list. This analysis provides an insight into the inherent difficulty of using the existing high-dimensional outlier detection techniques to solve the new outlying subspace detection problem.

A. HighDoD

Zhang *et al.* proposed a novel dynamic subspace search algorithm, called HighDoD, to efficiently identify the outlying subspaces for the given query data points [121][126]. The outlying measure, *OD*, is based on the sum of distances between a data and its k nearest neighbors [10]. This measure is simple and independent of any underlying statistical and distribution characteristics of the data points. The following two heuristic pruning strategies employing upward-and downward closure property are proposed to aid in the search for outlying subspaces: *If a point p is*

not an outlier in a subspace s, then it cannot be an outlier in any subspace that is a subset of s. If a point p is an outlier in a subspace s, then it will be an outlier in any subspace that is a superset of s. These two properties can be used to quickly detect the subspaces in which the point is not an outlier or the subspaces in which the point is an outlier. All these subspaces can be removed from further consideration in the later stage of the search process. A fast dynamic subspace search algorithm with a sample-based learning process is proposed. The learning process aims to quantitize the prior probabilities for upward- and downward pruning in each layer of space lattice. The *Total Saving Factor* (TSF) of each layer of subspaces in the lattice, used to measure the potential advantage in saving computation, is dynamically updated and the search is performed in the layer of lattice that has the highest TSF value in each step of the algorithm.

However, HighDoD suffers the following major limitations. First, HighDoD relies heavily on the closure (monotonicity) property of the outlying measurement of data points, termed OD, to perform the fast bottom-up or top-down subspace pruning in the space lattice, which is the key technique HighDoD utilizes for speeding up subspace search. Under the definition of OD, a subspace will always be more likely to be an outlying subspace than its subset subspaces. This is because that OD of data points will be naturally increased when the dimensionality of the subspaces under study goes up. Nevertheless, this may not be a very accurate measurement. The definition of a data point's outlier-ness makes more sense if its measurement can be related to other points, meaning that the averaged level of the measurement for other points in the same subspace should be taken into account simultaneously in order to make the measurement statistically significant. Therefore, the design of a new search method is desired in this situation. Secondly, HighDoD labels each subspace in a binary manner, either an outlying subspace or a non-outlying one, and most subspaces are pruned away before their outlying measurements are virtually evaluated in HighDoD. Thus, it is not possible for HighDoD to return a ranked list of the detected outlying subspaces. Apparently, a ranked list will be more informative and useful than an unranked one in many cases. Finally, a human-user defined cutoff

for deciding whether a subspace is outlying or not with respect to a query point is used. This parameter will define the "outlying front" (the boundary between the outlying subspaces and the non-outlying ones). Unfortunately, the value of this parameter cannot be easily specified due to the lack of prior knowledge concerning the underlying distribution of data point that maybe very complex in the high-dimensional spaces.

B. SOF Method

In [128], a novel technique based on genetic algorithm is proposed to solve the outlying subspace detection problem and well copes with the drawbacks of the existing methods. A new metric, called *Subspace Outlying Factor (SOF)*, is developed for measuring the outlying degree of each data point in different subspaces. Based on SOF, a new definition of outlying subspace, called SOF Outlying Subspaces, is proposed. Given an input dataset D, parameters n and k, a subspace s is a SOF Outlying Subspace for a given query data point p if there are no more than $n-1$ other subspaces s' such that $SOF(s', p) > SOF(s, p)$. The above definition is equivalent to say that the top n subspaces having the largest SOF values are considered to be outlying subspaces. The parameters used in defining SOF Outlying Subspaces are easy to be specified, and do not require any prior knowledge about the data distribution of the dataset. A genetic algorithm (GA) based method is proposed for outlying subspace detection. The upward and downward closure property is no longer required in the GA-based method, and the detected outlying subspaces can be ranked based on their fitness function values. The concepts of the lower and upper bounds of D^k, the distance between a given point and its k^{th} nearest neighbor, are proposed. These bounds are used for a significant performance boost in the method by providing a quick approximation of the fitness of subspaces in the GA. A technique is also proposed to compute these bounds efficiently using the so-called kNN Look-up Table.

2.3.3 Clustering Algorithms for High-dimensional Data

We have witnessed some recent developments of clustering algorithms towards high-dimensional data. As clustering provides a possible, even though not the best, means

to detect outliers, it is necessary for us to review these new developments. The representative methods for clustering high-dimensional data are CLIQUE and HPStream.

A. CLIQUE

CLIQUE [7] is a grid-based clustering method that discretizes the data space into non-overlapping rectangular units, which are obtained by partitioning every dimension into a specific number of intervals of equal length. A unit is dense if the fraction of total data points contained in this unit is greater than a threshold. Clusters are defined as unions of connected dense units within a subspace. CLIQUE first identifies a subspace that contains clusters. A bottom-up algorithm is used that exploits the monotonicity of the clustering criterion with respect to dimensionality: if a k-dimensional unit is dense, then so are its projections in (k-1) -dimensional space. A candidate generation procedure iteratively determines the candidate k-dimensional units C_k after determining the (k-1)-dimensional dense units D_{k-1}. A pass is made over the data to determine those candidates units that are dense D_k. A depth-first search algorithm is then used to identify clusters in the subspace: it starts with some unit u in D, assign it the first cluster label number, and find all the units it is connected to. Then, if there are still units in D that have yet been visited, it finds one and repeats the procedure. CLIQUE is able to automatically finds dense clusters in subspaces of high-dimensional dataset. It can produce identical results irrespective of the order in which input data are presented and not presume any specific mathematical form of data distribution. However, the accuracy of this clustering method maybe degraded due to the simplicity of this method. The clusters obtained are all of the rectangular shapes, which is obviously not consistent with the shape of natural clusters. In addition, the subspaces obtained are dependent on the choice of the density threshold. CLIQUE uses a global density threshold (*i.e.*, a parameter that is used for all the subspaces), thus it is difficult to specify its value especially in high-dimensional subspaces due to curse of dimensionality. Finally, the subspaces obtained are those where dense units exist, but this has nothing to do with the existence of outliers. As a result, CLIQUE is not suitable for detecting projected outliers.

B. HPStream

In order to find the clusters embedded in the subspaces of high-dimensional data space in data streams, a new clustering method, called HPStream, is proposed [9]. HPStream introduces the concept of *projected clustering* to data streams as significant and high-quality clusters only exist in some low-dimensional subspaces. The basic idea of HPStream is that it does not only find clusters but also updates the set of dimensions associated with each cluster where more compact clusters can be found. The total number of clusters obtained in HPStream is initially obtained through $k-$means clustering and the initial set of dimensions associated with each of these k clusters is the full set of dimensions of the data stream. As more streaming data arrive, the set of dimensions for each cluster evolves such that each cluster can become more compact with a smaller radius.

HPStream is innovative in finding clusters that are embedded in subspaces for high-dimensional data streams. However, the number of subspaces returned by HP-Stream is equal to the number of clusters obtained that is typically of a small value. Consequently, if HPStream is applied to detect projected outliers, then it will only be able to detect the outliers in those subspaces returned and miss out a significant potions of outliers existing in other subspaces that are not returned by HPStream. Of course, it is possible to increase the number of subspaces returned in order to improve the detection rate. However, the increase of subspaces will imply an increase of the number of clusters accordingly. An unreasonably large number of clusters is not consistent with the formation of natural clusters and will therefore affect the detection accuracy of projected outliers.

2.4 Outlier Detection Methods for Data Streams

The final major category of outlier detection methods we will discuss in this chapter are those outlier detection methods for handling data streams. We will first discuss Incremental LOF, and then the outlier detection methods for sensor networks that use Kernel density function. The incremental clustering methods that can handle continuously arriving data will also be covered at the end of this subsection.

A. Incremental LOF Method

Since LOF method is not able to handle data streams, thus an incremental LOF algorithm, appropriate for detecting outliers from dynamic databases where frequently data insertions and deletions occur, is proposed in [97]. The proposed incremental LOF algorithm provides an equivalent detection performance as the iterated static LOF algorithm (applied after insertion of each data record), while requiring significantly less computational time. In addition, the incremental LOF algorithm also dynamically updates the profiles of data points. This is an appealing property, since data profiles may change over time. It is shown that insertion of new data points as well as deletion of obsolete points influence only limited number of their nearest neighbors and thus insertion/deletion time complexity per data point does not depend on the total number of points N[97].

The advantage of Incremental LOF is that it can deal with data insertions and deletions efficiently. Nevertheless, Incremental LOF is not economic in space. The space complexity of this method is in the order of the data that have been inserted but have not been deleted. In other words, Incremental LOF has to maintain the whole length of data stream in order to deal with continuously arriving data because it does not utilize any compact data summary or synopsis. This is clearly not desired for data stream applications that are typically subject to explicit space constraint.

B. Outlier Detection Methods for Sensor Networks

There are a few recent anomaly detection methods for data streams. They mainly come from sensor networks domain such as [100] and [30]. However, the major effort taken in these works is the development of distributable outlier detection methods from distributed data streams and does not deal with the problem of outlier detection in subspaces of high-dimensional data space. Palpanas *et al.* proposed one of the first outlier detection methods for distributed data streams in the context of sensor networks [100]. The author classified the sensor nodes in the network as the low capacity and high capacity nodes, through which a multi-resolution structure of the sensor network is created. The high capacity nodes are nodes equipped with relatively strong computational strength that can detect local outliers. The Kernel density

function is employed to model local data distribution in a single or multiple dimensions of space. A point is detected as an outlier if the number of values that have fallen into its neighborhood (delimited by a sphere of radius r) is less than an application-specific threshold. The number of values in the neighborhood can be computed by the Kernel density function. Similarly, the authors in [30] also emphasize the design of distributed outlier detection methods. Nevertheless, this work employs a number of different commonly used outlier-ness metric such as the distance to k^{th} nearest neighbor, average distance to the k nearest neighbors, the inverse of the number of neighbors within a specific distance. Nevertheless, these metrics are not applicable to data streams.

C. Incremental Clustering Methods

Most clustering algorithms we have discussed earlier in this chapter assume a complete and static dataset to operate. However, new data becomes continuously available in many applications such as the data streams. With the aforementioned classical clustering algorithms, reclustering from scratch to account for data updates is too costly and inefficient. It is highly desired that the data can be processed and clustered in an incremental fashion. The recent representative clustering algorithms having mechanisms to handle data updates are BIRCH*, STREAM and CluStream.

BIRCH* [54] is a framework for fast, scalable and incremental clustering algorithms. In the BIRCH* family of algorithms, objects are read from the databases sequentially and inserted into incrementally evolving clusters which are represented by generalized cluster features (CF*s), the condensed and summarized representation of clusters. A new objects reading from the databases is inserted into the closest cluster. BIRCH* organizes all clusters in an in-memory index, and height-balanced tree, called CF*-tree. For a new object, the search for an appropriate cluster requires time logarithmic in the number of the clusters to a linear scan. CF*s are efficient because: (1) they occupy much less space than the naive representation; (2) the calculation of inter-cluster and intra-cluster measurements using the CF* is much faster than calculations involving all objects in clusters. The purpose of the CF*-tree is to direct a new object to the cluster closest to it. The non-leaf and leaf entries function

differently, non-leaf entries are used to guide new objects to appropriate leaf clusters, whereas leaf entries represent the dynamically evolving clusters. However, clustering of high-dimensional datasets has not been studied in BIRCH*. In addition, BIRCH* cannot perform well when the clusters are not spherical in shape due to the fact that it relies on spherical summarization to produce the clusters.

STREAM [92] considers the clustering of continuously arriving data, and provides a clustering algorithm superior to the commonly used k-means algorithm. STREAM assumes that the data actually arrives in chunks $X_1, X_2, \cdots, X_n$, each of which fits into main memory. The streaming algorithm is as follows. For each chunk i, STREAM first assigns weight to points in the chunks according to their respective appearance frequency in the chunks ensuring that each point appear only once. The STREAM clusters each chunk using procedure LOCALSEARCH. For each chunk, only k weighted cluster centers are retained and the whole chunk is discarded in order to free the memory for new chunks. Finally, LOCALSEARCH is applied to the weighted centers retained from $X_1, X_2, \cdots, X_n$, to obtain a set of (weighted) centers for the entire stream $X_1, X_2, \cdots, X_n$.

In order to find clusters in different time horizons (such as the last month, last year or last decade), a new clustering method for data stream, called CluStream, is proposed in [8]. This approach provides the user the flexibility to explore the nature of the evolution of the clusters over different time periods. In order to avoid bookkeeping the huge amount of information about the clustering results in different time horizons, CluStream divides the clustering process into an online micro-clustering component and an offine macro-clustering component. The micro-clustering phase mainly collects online the data statistics for clustering purpose. This process is not dependent on any user input such as the time horizon or the required granularity of the clustering process. The aim is to maintain statistics at a sufficiently high level of granularity so that it can be effectively used by the offline components of horizon-specific macro-clustering as well as evolution analysis. The micro-clusters generated by the algorithm serve as an intermediate statistical representation which can be maintained in an efficient way even for a data stream of large volume. The macro-clustering process

does not work on the original data stream that may be very large in size. Instead, it uses the compactly stored summary statistics of the micro-clusters. Therefore, the micro-clustering phase is not subject to the one-pass constraint of data stream applications.

D. Advantages and Disadvantages of Outlier Detection for Data Streams

The methods discussed in this subsection can detect outliers from data streams. The incremental LOF method is able to deal with continuously arriving data, but it may face an explosion of space consumption. Moreover, the incremental LOF method is not able to find outliers in subspaces in an automatic manner. The outlier detection methods for sensor networks cannot find projected outliers either. Unlike the clustering methods that are only appropriate for static databases, BIRCH*, STREAM and CluStream go one step further and are able to handle incrementally the continuously arriving data. Nevertheless, they are designed to use all the features of data in detecting outliers and are difficult to detect projected outliers.

2.5 Summary

This section presents a review on the major existing methods for detecting point outliers from vector-like data sets. Both the conventional outlier detection methods that are mainly appropriate for relatively low dimensional static databases and the more recent methods that are able to deal with high-dimensional projected outliers or data stream applications have been discussed. For a big picture of these methods, we present a summary in Table 2.6. In this table, we evaluate each method against two criteria, namely whether it can detect projected outliers in a high-dimensional data space and whether it can handle data streams. The symbols of tick and cross in the table indicate respectively whether or not the corresponding method satisfies the evaluation criteria. From this table, we can see that the conventional outlier detection methods cannot detect projected outliers embedded in different subspaces; they detect outliers only in the full data space or a given subspace. Amongst these methods that can detect projected outliers, only HPStream can meet both criteria. However, being a clustering method, HPStream cannot provide satisfactory support for projected

outliers detection from high-dimensional data streams, as we have discussed in details in Subsection 2.2.4. Thus, a new technique that is able to detect projected outliers from high-dimensional data streams is desired.

Category	Method	High-D subspace outliers	Data Stream
Statistical detection methods	Gaussian models	X	X
	Regression models	X	√
	Histograms	X	√
	Kernel functions	X	√
Distance-based methods	DB(k, λ)-Outliers	X	X
	DB(pct, d_{min})-Outliers	X	X
	K^{th} NN method	X	X
	K^{th} NN sum method	X	X
	Grid-ODF	X	X
Density-based methods	LOF	X	X
	COF	X	X
	INFLO	X	X
	MDEF	X	X
	Incremental LOF	X	√
Clustering-based methods	PAM/CLARA	X	X
	k-means	X	X
	CLARANS	X	X
	MST clustering	X	X
	BIRCH	X	√
	CURE	X	X
	CHAMELEON	X	X
	DBSCAN	X	X
	DENCLUE	X	X
	STING	X	√
	WaveCluster	X	√
	DCLUST	X	√
	STREAM	X	√
	BIRCH*	X	√
	CluStream	X	√
High-D outlier detection methods	Sparse Coefficient method	√	X
	Example-based method	√	X
	HighDoD	√	X
	SOF method	√	X
High-D clustering-based methods	CLIQUE	√	X
	HPStream	√	√

Figure 2.6: A summary of major existing outlier detection methods

Chapter 3

Concepts and Definitions

In this section, we will cover the basic concepts and definitions that are used in our technique. First, we will introduce the time model used to distinguish data in different time horizons. The data synopsis for compactly capturing sufficient statistical information for outlier detection are then proposed. Definition of projected outliers in subspaces for high-dimensional data streams is also presented. Finally, detailed discussions are given on how to compute and maintain the data synopsis efficiently in order to meet the requirements of data stream applications.

3.1 Time Model and Decaying Function

We use a novel window-based time model, called (ω, ϵ)-model, in SPOT for discriminating data arriving at different times in the stream. Unlike the conventional window-based model, (ω, ϵ)-model does not need to keep trace of all the detailed data in the window. Moreover, instead of maintaining a large number of historical snapshots of data synopsis as in the tilted time models, only the latest snapshot needs to be kept in the (ω, ϵ)-model.

The concept of a sliding window is used in (ω, ϵ)-model. The parameter ω is the window size representing either the difference in terms of time (or stream length if no timestamps are present) between the first and last data located in the sliding window W, *i.e.*,

$$\forall p_i \in W, T - T_i \leq \omega \text{ or } N - N_i \leq \omega$$

where T and T_i denotes the current time and the arrival time of p_i, N and N_i correspond to the current stream length and that when p_i arrived. ω can be defined, for example, as $\omega = 10$ minutes or $\omega = 1000$ data, meaning that we are only interested in the data arriving less than 10 minutes ago or the recent 1000 arriving data. For

ease of presentation, we define ω based on the dimension of time in the reminder of this thesis.

In (ω, ϵ)-model, each data in the window will be assigned a weight, indicating its importance or influence to the data synopsis at the current time. We use *exponential decaying function* for calculating weights for different data. It is a function of the elapsed time from the current time and the time when the data arrived. Precisely, suppose p_i arrived at time T_i and the current time is T, then weight of p_i is computed as

$$weight(p_i) = df(T - T_i) = e^{-\frac{\alpha(T-T_i)}{t}}$$

where α is the decaying coefficient that is used to adjust the speed of weighted decay, and t is the basic time unit used for scaling elapsed time in decaying function.

The definition of (ω, ϵ)-model is given as follows.

Definition 3.1 (ω, ϵ)**-model**: For the points that have slid out of the current window W (with a size of ω), the sum of their weights will not exceed ϵ. Formally, we have $\sum_i weight(p_i) \leq \epsilon$, where $T - T_i > \omega$, T is the current time and T_i denotes the arriving time for point p_i. The (ω, ϵ)-model is an approximation of conventional window-based model of a window size ω with an approximation factor of ϵ.

In SPOT, we take advantage of the flexibility offered by the decaying function to implement the (ω, ϵ)-model, without maintaining detailed data in the sliding window. The basic idea is to tune the decaying coefficient α such that the influence from those data that have slid out the window could be negligible. We explore the specification of decaying coefficient α in Lemma 3.1.

Lemma 3.1. Let T_{min} be the minimum interval of the arriving time for two consecutive data in the stream. At any time T, if the decaying coefficient α is set such that $e^{\alpha\frac{\omega}{t}}(e^{\alpha\frac{T_{min}}{t}} - 1) \geq \frac{1}{\epsilon}$, then $\sum_i weight(p_i) \leq \epsilon$, where $T - T_i > \omega$.

Proof. Suppose the data that have slid out of the current window, starting from the more recent ones, are p_1, p_2, p_3, $\cdots$. Since for any data p_i, we have $weight(p_i) \leq e^{-\alpha\frac{\omega+i\cdot T_{min}}{t}}$, thus $\sum_i weight(p_i)$ can be bounded as follows:

$$\sum_i weight(p_i) \leq e^{-\alpha\frac{\omega+T_{min}}{t}} + e^{-\alpha\frac{\omega+2T_{min}}{t}} + e^{-\alpha\frac{\omega+3T_{min}}{t}} + \cdots \tag{3.1}$$

Since $e^{-\alpha\frac{\omega+T_{min}}{t}} + e^{-\alpha\frac{\omega+2T_{min}}{t}} + e^{-\alpha\frac{\omega+3T_{min}}{t}} + \cdots$ is the sum of an infinite decreasing geometric series, thus Eq. (3.1) can be simplified as

$$\sum_i weight(p_i) \leq \frac{e^{-\alpha\frac{\omega+T_{min}}{t}}}{1 - e^{-\alpha\frac{T_{min}}{t}}}$$

In order to satisfy $\sum_i weight(p_i) \leq \epsilon$, we need to have $\frac{e^{-\alpha\frac{\omega+T_{min}}{t}}}{1-e^{-\alpha\frac{T_{min}}{t}}} \leq \epsilon$. By simplifying this, we can get $e^{\alpha\frac{\omega}{t}}(e^{\alpha\frac{T_{min}}{t}} - 1) \geq \frac{1}{\epsilon}$, as required. **(End of proof)**

Lemma 3.1 suggests that, by choosing an appropriate decay coefficient α (no less than α^*, where α^* is the root of equation $e^{\alpha\frac{\omega}{t}}(e^{\alpha\frac{T_{min}}{t}} - 1) = \frac{1}{\epsilon}$), we can implement window-based time model without keeping trace of data in the window or maintaining multiple snapshots of data synopsis. This contributes to a better computational and space performance.

Using the decaying function, the data synopsis at time T' can be defined recursively by the latest snapshot of data synopsis created at time T, $T \leq T'$, as

$$\mathcal{M}_{T'} = df(T' - T)\mathcal{M}_T + p_{T'}$$

where the additive term p_T denotes the summary of those data arrived at time T'.

There are other types of decaying functions such as the linear decaying functions, but the results obtained using the exponential decaying function cannot be generalized to the linear decaying function. Compared with an exponential decaying function, a linear decaying function tends to have a slower decaying speed and the aggregated sum of the weights of the data that have slid out of the sliding window is not upper bounded, which violates the definition of our (ω, ϵ) time model.

Please note that, in the (ω, ϵ) time model, we assume that there is a minimum interval of the arriving time for two consecutive data in the stream, denoted by T_{min}. If multiple data arrive at the same time, SPOT still processes them sequentially. Therefore, the assumption of minimum time interval between two consecutive data still hold.

A similar decaying function was used in [8][9]. However, it does not use the scaling factor t and the decaying factor α is not fine tuned explicitly for implementing the window-based model.

3.2 Data Synopsis

Like many other data mining tasks on data stream analysis such as clustering and frequent item detection, one of central problems involved is the design of appropriate synopsis over data streams that are suitable for outlier detection purpose. In our work, we employ *Base Cell Summary* (BCS) and *Projected Cell Summary* (PCS), two compact structures that are able to capture the major underlying characteristics of the data stream for detecting projected outliers.

Quantitization of BCS and PCS entails an equi-width partition of the domain space, which partitions each attribute into a few disjoint subsets, or buckets. Specifically, let $\mathcal{D}$ be a set of φ-dimensional streaming data objects. For each attribute, the space is partitioned into a set of non-overlapping intervals. All the intervals in each such dimension are of identical length. The cells in hypercube can be classified into two categories, *i.e.*, the base and projected cells. A *base cell* is a cell in hypercube with the finest granularity. The dimensionality (*i.e.*, number of attributes) of base cells is equal to φ. A *projected cell* is a cell that exists in a particular subspace s, which is a projection of a number of base cells from the full data space into s. The dimensionality of a projected cell is equal to $|s|$ and $|s| < \varphi$, where $|s|$ denotes the number of attributes in s. For a simple example, if each dimension of a 3-dimensional data space is divided into 3 intervals, then there are 27 base cells, but only 9 projected cells in all 2-dimensional subspaces and 3 projected cells in all 1-dimensional subspaces.

Definition 3.2 Base Cell Summary (BCS): The Base Cell Summary of a base cell c in the hypercube is a triplet defined as $BCS(c) = \{D_c, \vec{LS}_c, \vec{SS}_c)$, where D_c, $\vec{LS}_c$ and $\vec{SS}_c$ denote the number of points, the sum and squared sum of data values in each dimension of points in c, respectively, *i.e.*, $\vec{LS}_c = \sum \vec{p}_i$ and $\vec{SS}_c = \sum \vec{p}_i^{\,2}$, for p_i located in c, $1 \leq i \leq \varphi$. n_c is a scalar while both $\vec{LS}_c$ and $\vec{SS}_c$ are φ-dimensional

vectors.

BCS features the following two desirable properties, *i.e.*, additivity and incremental maintainability [129], that can be used to compute data synopsis for projected cells in subspaces.

Property 1. **Additivity of BCS**: Let $BCS(c_1) = (D_{c_1}, \vec{LS}_{c_1}, \vec{SS}_{c_1})$ and $BCS(c_2) = (D_{c_2}, \vec{LS}_{c_2}, \vec{SS}_{c_2})$ be BCSs of two base cells c_1 and c_2, respectively. It is easy to see that the $D_{c'}$, $\vec{LS}_{c'}$ and $\vec{SS}_{c'}$ of the cell c' that is formed by the union of c_1 and c_2 can be presented as [129]:

$$BCS(c') = (D_{c'}, \vec{LS}_{c'}, \vec{SS}_{c'}) = (D_{c_1}+D_{c_2}, \vec{LS}_{c_1} + \vec{LS}_{c_2}, \vec{SS}_{c_1} + \vec{SS}_{c_2}) = BCS(c_1)+BCS(c_2)$$

Property 2: **Incremental maintainability of BCS**: Suppose that the latest snapshot of BCS of c is created at time T, then we can update BCS of c when a new data point p arrives at c at the time T' ($T \leq T'$) as follows:

$$BCS(c,s)^{T'} = df(T'-T)BCS(c,s)^{T} + [1, \vec{p}, \vec{p^2}]$$

Definition 3.3 Projected Cell Summary (PCS): The Projected Cell Summary of a cell c in a subspace s is a scalar triplet defined as $PCS(c,s) = (RD, IRSD, IkRD)$, where RD, $IRSD$ and $IkRD$ are the Relative Density, Inverse Relative Standard Deviation and Inverse k-Relative Distance of data points in c of s, respectively.

RD, $IRSD$ and $IkRD$ are three effective measures to represent the overall data sparsity of each projected cell from different perspectives. They are used together in SPOT to achieve a good measurement of data outlier-ness. They are all defined as ratio-type measures in order to achieve statistical significance in measurement and facilitate the specification of the outlier-ness thresholds. They can be computed and updated incrementally and are thus suitable for data stream applications. In what follows, we will elaborate on these three components of PCS.

3.2.1 Relative Density (RD)

Relative Density of a cell c in subspace s measures the relative density of c w.r.t the expected level of density of non-empty cells in s. If the density of c is significantly

lower than the average level of cell density in the same subspace, then the data in c can be labeled as outliers. It is defined as $RD(c,s) = \frac{D_c}{E(D_s)}$, where D_c and $E(D_s)$ represent the density (*i.e.*, number of points) in c and the expected density of all the cells in s. Since $E(D_s) = \frac{N_\omega}{\delta^{|s|}}$, where N_ω corresponds to the effective stream length (the decayed number of data points in sliding window at a certain time), thus,

$$RD(c,s) = \frac{D_c \cdot \delta^{|s|}}{N_\omega} \tag{3.2}$$

3.2.2 Inverse Relative Standard Deviation (IRSD)

Inverse Relative Standard Deviation of a cell c in subspace s is defined as inverse of the ratio of standard deviation of c in s against the expected level of standard deviation of non-empty cells in s. Under a fixed density, if the data in a cell features a remarkably high standard deviation, then the data are generally distributed more sparsely in the cell and the overall outlier-ness of data in this cell is high.

$IRSD(c,s)$ is computed as $IRSD(c,s) = \left[\frac{\sigma_c}{E(\sigma_s)}\right]^{-1}$, where σ_c denotes the standard deviation of c and $E(\sigma_s)$ denotes the expected standard deviation of cells in subspace s. Since σ_c is larger than 0 but does not exceed the length of the longest diagonal of the cell in subspace s, which is $\sqrt{|s|}l$, where l is the side length of each interval of the cell, thus $E(\sigma_s)$ can be estimated as $E(\sigma_s) = \frac{0+\sqrt{|s|}l}{2} = \frac{\sqrt{|s|}l}{2}$. Thus, IRSD can be expressed as

$$IRSD(c,s) = \frac{\sqrt{|s|}l}{2\sigma_c} \tag{3.3}$$

Remarks: Notice that $|s|$ appears in the mathematical definition of both RD and IRSD. It serves as a normalization factor to make quantities of RDs (and IRSDs) across different subspaces in a comparable magnitude. **(End of remarks)**

3.2.3 Inverse *k*-Relative Distance (IkRD)

Inverse k-Relative Distance for a cell c in a subspace s is the inverse of ratio of the distance between the centroid of c and its nearest representative points in s against

the average level of such distance in s for all the non-empty cells. A high IkRD value of c indicates that c is noticeably far from the dense regions of the data in s, thus the outlier-ness of data in c is high. The notion of representative points will be discussed in Subsection 3.4.4.

Given parameter k, the IkRD of a projected cell c in subspace s is defined as

$$IkRD(c, s, k) = \left[\frac{k\_dist(c, s)}{average\_k\_dist(c_i, s)}\right]^{-1}$$

where $k\_dist(c, s)$ is the sum of distances between the centroid of c and its k nearest representative points in s. *i.e.*,

$$k\_dist(c, s) = \sum dist(centroid(c), r_i, s), r_i \in kNN(c, s)$$

$kNN(c, s)$ is the set containing the k nearest representative points of c in s.

$average\_k\_dist(c_i, s)$ is the average level of $k\_dist(c_i, s)$ for all the non-empty cells in s, which is computed as

$$average\_k\_dist(c_i, s) = \frac{\sum k\_dist(c_i, s)}{|c_i|}, c_i \in s$$

Remarks: Note that the inverse of IRSD and IkRD, rather than RSD and kRD, are used in SPOT so as to transform the problem of finding outlying subspaces to a minimization problem. A small PCS (containing RD, IRSD and IkRD) indicates a high level of data outlier-ness of c in s and vice versa. **(End of remarks)**

3.3 Definition of Projected Outliers

We base our decision of data points' outlier-ness on the data sparsity of the cells they belong to and the distance between them to the representative points in the subspace. Specifically, a data point in a subspace is regarded as outlying if its RD, IRSD or IkRD levels becomes unusually lower than the average/expected level. The outlying cell of a point is defined as follows.

Definition 3.4 outlying cell: A cell c in subspace s is called the outlying cell of a point p in s if one or more components of $PCS(c, s)$ (*i.e.,* RD, IRSD or IkRD) exceed their corresponding thresholds, where p is located in c.

As we know that each streaming point can be only mapped into one and only one cell in the hypercube, thus there is at most one outlying cell of a point in any subspace, if any. Based on the concept of outlying cell of a point, we can define outlying subspace of a point.

Definition 3.5 outlying subspace: An outlying subspace s of a point p is a subspace in which p is an outlier in a particular cell of s. In other words, an outlying subspace s of p is a subspace that contains the outlying cell of p.

Finally, based on concept of outlying subspace, we can formally define the notion of projected outliers.

Definition 3.6 Projected outliers: A data point p is considered as a projected outlier if there exists *at least* one outlying subspace s of p.

Projected outliers are important entities as outliers that exist in a high dimensional space are projected outliers. They carry abnormal behavior or patterns in a subset of attributes of the data. The aim of SPOT is to screen out projected outliers from continuously arrived data points. Unless otherwise stated, the outliers discussed in SPOT refer to projected outliers.

3.4 Computing PCS of a Projected Cell

In this subsection, we will discuss in details how PCS of a projected cell in the hypercube can be quantitized. A promising characteristics of PCS is that, once BCS of all the base cells in hypercubes have been calculated, PCS of any projected cell can be computed efficiently by only aggregating the BCS of its base cells, without referring to any detailed data points. This ensures that the computation of PCS will only needs a single scan of the data stream.

Definition 3.7 Base cell set of a projected cell: Let c be a projected cell in subspace s with attribute set $[a_1, a_2, \ldots, a_{|s|}]$. The interval index numbers of the $|s|$ attributes of c can be presented as $[I_1, I_2, \ldots, I_{|s|}]$. Similarly, the attribute set and the corresponding attribute interval index of a base cell can be presented as $[a'_1, a'_2, \ldots, a'_\varphi]$ and $[I'_1, I'_2, \ldots, I'_\varphi]$, respectively. The base cell set of c, denoted as $Base\_Cell\_Set(c)$,

can be presented as

$$Base\_Cell\_Set(c) = \{c',\ \text{where } |c'| = \varphi \text{ and } |I_i = I'_j| = |s|\}, 1 \leq i \leq |s|, 1 \leq j \leq \varphi$$

$|I_i = I'_j| = |s|$ signifies that the number of attributes whose index number are identical between c and c' is equal to $|s|$. The rationale for this definition is the fact that, between a project cell and any one of its base cells, there exists exactly $|s|$ attributes of them that would have the same index number.

The central components needs to be obtained for RD, IRSD, IkRD of PCS of a projected cell c are the density (D_c), mean (μ_c) and standard deviation (σ_c) of data points in c and the representative points in a given subspace s. In the following, we will focus on how D_c, μ_c, σ_c and representative points can be obtained thanks to the additivity property of BCS.

3.4.1 Computing Density of a Projected Cell D_c

Let c be a projected cell in subspace s. The density of c can be computed as follows:

$$D_c = \sum_i D_{c'_i},\ \text{where } c'_i \text{ is in Base\_Cell\_Set(c)} \tag{3.4}$$

The densities of base cells of c' are pre-computed, so D_c can be computed efficiently.

3.4.2 Computing Mean of a Projected Cell μ_c

Let c be a projected cell in subspace s that contains m points. The mean of c is computed as follows:

$$\vec{\mu_c} = \frac{\sum_{i=1}^{m} \vec{p_i}}{m} = \frac{1}{m}[\sum_i^m p_i(1), \sum_i^m p_i(2), \ldots, \sum_i^m p_i(|s|)] \tag{3.5}$$

where $\vec{p_i}$ denotes the vector representation of i^{th} data point in c ($1 \leq i \leq m$) and $p_i(j)$ represents the j^{th} attribute of point p_i ($1 \leq j \leq \varphi$). Each element $\sum_i^m p_i(j)$ can be computed as $\sum_i^m p_i(j) = \sum_{p'} p'(j)$, where p' is located in Base_Cell_Set(c). Since $\sum_{p'} P'(j)$ is available in the BCSs of base cells of c, thus μ_c can be directly computed by BCSs of base cells of c.

3.4.3 Computing Standard Deviation of a Projected Cell σ_c

The unbiased estimator of standard deviation of m data points in c is computed as

$$\sigma_c = \sqrt{\frac{\sum_{i=1}^{m} Dist(p_i, \mu_c)^2}{m-1}} \tag{3.6}$$

with a degree of freedom of $m-1$, where $Dist(p_i, \mu_c)$ is the distance between point p_i and μ_c, and $Dist(p_i, \mu_c)^2$ can be computed as

$$Dist(p_i, \mu_c)^2 = \sum_{j=1}^{|s|} [p_i(j)^2 - 2p_i(j)\mu_c(j) + \mu_c(j)^2] \tag{3.7}$$

Plug Eq. (3.7) into Eq. (3.6), we can get

$$\sigma_c = \sqrt{\frac{\sum_{j=1}^{|s|} [\sum_{i=1}^{m} p_i(j)^2 - 2\mu_c(j) \sum_{i=1}^{m} p_i(j) + m\mu_c(j)^2]}{m-1}} \tag{3.8}$$

where $\sum_i^m p_i(j) = \sum_{p'} p'(j)$, $\sum_i^m p_i(j)^2 = \sum_{p'} p'(j)^2$, p' is a data located in *Base_Cell_Set(c)*. Because both $\sum_{p'} p'(j)$ and $\sum_{p'} p'(j)^2$ are available in the BCSs of vase cells of c, thus, like the mean of points in c, its standard deviation can also be directly computed by BCSs of its base cells.

3.4.4 Generate Representative Data Points in a Subspace

The representative points of a subspace needs to be generated to compute $IkRD$ of cells in this subspace. These representative points are selected from the centroids of non-empty cells in the subspace. This set of selected cells are termed as *coverage cells*. A rule-of-thumb in selecting coverage cells in a subspace s is to select a specified number of the most populated cells in s such that they cover a majority of data in s. Specifically, the cells in s will be first sorted in descending order based on their densities. Then, starting from the most populated cell(s), $N_{Coverage\_cell}(s)$ cells are chosen such that the coverage of these $N_{Coverage\_cell}(s)$ cells is no less than q $(0 \leq q \leq 1)$ of the total data in s, where $N_{Coverage\_cell}(s)$ denotes the number of coverage cells to be selected and q is termed *coverage ratio* that is expected to take a value close to 1, such as 0.9 or 0.95. Mathematically, $N_{Coverage\_cell}(s)$ can be expressed as

$$N_{Coverage\_cell}(s) = t, where \sum_{i=1}^{t-1} density(c_i) \leq qN_\omega \leq \sum_{i=1}^{t} density(c_i)$$

To compute IkRD of a subspace, we need to maintain the following two properties during the process of data streams:

- The centroid/mean of all populated cells in this subspace;
- The average distance between the centroid of a cell and its k nearest representative points for populated cells in this subspace.

Based on the discussions in Subsection 3.4.1 and 3.4.2, we know that these two information can be easily obtained from BCS. It is noted that the initial representative data for different subspaces are obtained offline before outlier detection is conducted, thus it is not subject to the one-pass scan and time-criticality requirements of data stream applications.

3.5 Maintaining the PCS of a Projected Cell

Besides computing PCSs of projected cells, another important issue will be how to maintain these synopses efficiently in the detection stage to meet the needs of data stream applications. The naive approach for doing this is to first update BCS of base cells when new data arrives and then project base cells from the full domain space into different subspaces in order to update their PCSs. This approach, however, will be expensive because as high as $\delta^{\varphi-|s|}$ aggregations may be involved. Moreover, the PCS of projected cells has to be updated in this way each time when a new point arrives at this cell, thus frequent aggregation operations are required, which makes this naive method rather costly. To achieve an efficient maintenance of the PCS, we will demonstrate the incremental property of the PCS in this subsection. Once it have been computed, the PCS can be thereafter updated incrementally upon the arrival of each subsequent data. This enables the maintenance of the PCS to be performed efficiently, without even referring to BCS of its base cells any more. This can help avoid the need to aggregate BCS for updating the PCS and thus leads to a computation saving by a large magnitude.

3.5.1 Update RD of the PCS

Suppose that the latest snapshot of the PCS of a cell c is created at time T with an effective stream length N_ω, then we can update RD in the PCS of c incrementally when a new data point arrives at c at the time T' with an effective stream length N'_ω as follows:

$$RD(c,s)^{T'} = \frac{[df(T'-T)\frac{RD(c,s)^T \cdot N_\omega}{\delta^{|s|}} + 1] \cdot \delta^{|s|}}{N'_\omega} \tag{3.9}$$

where $\frac{RD(c,s)^T \cdot N_\omega}{\delta^{|s|}}$ is the density of c at time T and $df(T'-T)\frac{RD(c,s)^T \cdot N_\omega}{\delta^{|s|}} + 1$ is thus the new density of c at time T'. After simplifying Eq. (3.9), we can get

$$RD(c,s)^{T'} = df(T'-T)\frac{N_\omega}{N'_\omega}RD(c,s)^T + \frac{\delta^{|s|}}{N'_\omega} \tag{3.10}$$

From Eq. (3.10) we can see that, for incremental maintenance of the PCS, we need to only additionally maintain, for each populated cell in the subspace, the effective stream length N_ω when the PCS of this cell was updated last time.

Before we start off discussing the update of IRSD in the PCS, we need to first present the following lemma.

Lemma 3.2. As the density of a projected cell increases, the mean of this cell will tend to converge. Mathematically, let the density of this projected cell is m, given ϵ, $\exists n > 0$, when $m \geq n$, we will have

$$\frac{Dist(\mu_c^{m+1}, \mu_c^m)}{\sqrt{|s|}L} \leq \epsilon$$

where ϵ is a small positive real number.

Proof: let μ_c^m be the mean of cell c when it contains m and μ_c^{m+1} be the new mean of this cell when a new point is added. The distance between μ_c^m and μ_c^{m+1} is computed as

$$Dist(\mu_c^{m+1}, \mu_c^m) = \sqrt{\sum_{i=1}^{|s|}(\mu_c^{m+1}(i) - \mu_c^m(i))^2}$$

$\mu_c^m(i)$ be the i^{th} dimension of the mean of m data points in cell c.

Since $\mu_c^{m+1}(i) = \frac{m\mu_c^m(i) + p_{m+1}(i)}{m+1}$, $p_{m+1}(i)$ is the i^{th} dimension of the $(m+1)^{th}$ poin in c, thus

$$(\mu_c^{m+1}(i) - \mu_c^m(i))^2 = (\frac{m\mu_c^m(i) + p_{m+1}(i)}{m+1} - \mu_c^m(i))^2$$

$$= \frac{(p_{m+1}(i) - \mu_c^m(i))^2}{(m+1)^2}$$

Since $0 \leq (p_{m+1}(i) - \mu_c^m(i))^2 \leq L^2$, thus $(\mu_c^{m+1}(i) - \mu_c^m(i))^2 \leq \frac{L^2}{(m+1)^2}$. We then have

$$Dist(\mu_c^{m+1}, \mu_c^m) \leq \frac{\sqrt{\sqrt{|s|} \cdot \frac{L^2}{(m+1)^2}}}{\sqrt{|s|}L} = \frac{1}{m+1}$$

Since $\frac{1}{m+1}$ converges to 0 when $m \to +\infty$, thus $\exists n > 0$, when $m \geq n$, we will have

$$\frac{Dist(\mu_c^{m+1}, \mu_c^m)}{\sqrt{|s|}L} \leq \epsilon$$

as required. (**End of proof**)

Lemma 3.2 reveals that, when the density of c is getting large, the mean μ of c is asymptotically stabilized. This enable us to reasonably consider the mean of cell as fixed when we update the PCS of the cells. This lemma is quite useful for accomplishing the update of RSD of c in an incremental manner.

3.5.2 Update IRSD of the PCS

Suppose the density of cell c is m and let $IRSD(c,s)^T$ be the IRSD of cell c in subspace s at time T, which can be computed as follows based on the definition of $IRSD(c,s)^T$:

$$IRSD(c,s)^T = \frac{\sqrt{|s|}l}{2\sigma(c)} = \frac{\sqrt{|s|}l}{2}\sqrt{\frac{m-1}{\sum_{i=1}^m Dist(p_i, \mu_c)^2}} \tag{3.11}$$

where p_i is located in c. Based on Eq. (3.11), we can get

$$\sum_{i=1}^m Dist(p_i, \mu_c)^2 = \frac{|s|l^2(m-1)}{4(IRSD(c,s)^T)^2} \tag{3.12}$$

The $IRSD(c,s)$ after the $(m+1)^{th}$ point is assigned into c at time T' ($T \leq T'$) is computed as

$$IRSD(c,s)^{T'} = \frac{\sqrt{|s|}l}{2}\sqrt{\frac{df(T'-T)(m-1)+1}{df(T'-T)\sum_{i=1}^m Dist(pi, \mu_c')^2 + Dist(p_{m+1}, \mu_c')^2}} \tag{3.13}$$

where μ_c' denotes the *new* mean of points in c when the $(m+1)^{th}$ point is inserted into c. Based on the Lemma 3.2, we can approximate $IRSD(c,s)^{T'}$ by plugging Eq. (3.12) into Eq. (3.13) as

$$IRSD(c,s)^{T'} = \sqrt{\frac{(df(T'-T)(m-1)+1)sl^2(IRSD(c,s)^T)^2}{df(T'-T)sl^2(m-1) + 4(IRSD(c,s)^T)^2 Dist(p_{m+1}, \mu_c')^2}} \tag{3.14}$$

Please note that, in order to compute $Dist(p_{m+1}, \mu'_c)^2$ in Eq. (3.14), we need to first update the mean of points in c and then compute the distance between p_{m+1} and the new mean μ'_c.

3.5.3 Update Representative Points in a Subspace

A crucial task in ensuring a fast update of IkRD is to obtain the set of coverage cells efficiently in each subspace of SST when a data is processed in the stream. A native way would be to re-sort all the populated cells in the subspace and pick up a specific number of the top cells as the new coverage cells in this subspace. This method is, however, expensive given the fact that the sorting has been performed in each subspace for each incoming data and the number of populated cells in some subspaces may be large.

A better way for maintaining the set of coverage cells is to reduce the number of times populated cells are sorted as much as possible; sorting may not be necessary each time when a new data arrives. The underlying rationale is that, in most cases, a small amount of newly arrived data may not significantly change the overall density distribution of data in the subspace and thus will not have any effect on the set of coverage cells. Thus, if we can make this certain, then sorting is definitely not necessary for the time being. To this end, we devise the following heuristics to minimize the number of re-sorting of projected cells:

1. If a new data falls into one of the coverage cells in a subspace, then there is no need to update the current set of coverage cells in this subspace. The proof of this heuristic is given in Lemma 3.3;

2. Both the total coverage and the minimum density of the current set of coverage cells in each subspace $s \in SST$ are maintained, denoted by Cov and Den_{min}, respectively. If a new data falls into a non-coverage cell c' in s, then there is no need to update the current set of coverage cells in s if we have $Cov' > q$ and $den(c') \leq Den'_{min}$, where Cov' and Den'_{min} correspond respectively to the

decayed Cov and Den_{min} after the new data is processed, and q denotes the coverage ratio required for the coverage cells. Both Cov' and Den'_{min} can be updated efficiently.

The reason that the current set of coverage cells does not need to update in the above two cases is that, after arrival of the new data, the current coverage cells still satisfy the coverage requirement and are still the cells with the highest densities in the subspace. These two heuristics can contribute to a significant efficiency improvement as the majority of the data are expected to fall into coverage cells. For example, if the coverage cells cover 95% of the data in a subspace, then the possibility that the coverage cells need to be updated when processing a new data is much lower than 5% in practice by using these two heuristics.

Lemma 3.3.: Let $CC^m(s)$ denote the set of coverage cells in subspace s after the m^{th} data points have been processed. If the $(m+1)^{th}$ data point p falls into a coverage cell, then $CC^{m+1}(s) = CC^m(s)$, meaning that coverage cells in s remain unchanged.
Proof: The coverage of cells in $CC^m(s)$ when the $(m+1)^{th}$ data point arrives is

$$(D(c', s) \times df(.) + 1) + \sum D(c_i, s) \times df(.),\ c' \in CC^m(s), c_i \in CC^m(s)$$

where c' is the coverage cell where the $(m+1)^{th}$ data point falls into, while c_i are other coverage cells.

The coverage of non-coverage cells (*i.e.*, the cells that are not in $CC^{m+1}(s)$) when the $(m+1)^{th}$ data point arrives is

$$\sum D(c_j, s) \times df(.)),\ c_j \notin CC^m(s)$$

Based on the coverage requirement for the coverage cells, we have

$$D(c', s) + \sum D(c_i, s) \geq q(D(c', s) + \sum D(c_i, s) + \sum D(c_j, s))$$

Therefore, we can get

$$D(c', s) \times df(.) + 1 + \sum D(c_i, s) \times df(.) \geq q(D(c', s) \times df(.) + \sum D(c_i, s + \sum D(c_j, s)) \times df(.)) + 1$$

$$> q(D(c', s) \times df(.) + 1 + \sum D(c_i, s) \times df(.) + \sum D(c_j, s) \times df(.))$$

suggesting the cells in $CC^m(s)$ still satisfy the coverage requirement after the $(m+1)^{th}$ data point arrives.

In addition, after the $(m+1)^{th}$ data point arrives, for any non-coverage cell c_j

$$D(c_j, s) \times df(.) \leq density_{min} \times df(.)$$

meaning that there is no non-coverage cells whose density can exceed that of any coverage cells. In other words, the current coverage cells are still the cells with the highest densities in s.

Based on above discussion, we can conclude that coverage cells in s remain unchanged after after the $(m+1)^{th}$ data point arrives, *i.e.*, $CC^{m+1}(s) = CC^m(s)$. (**End of proof**)

In a summary, to compute IkRD in SPOT, the following steps are performed in each subspace s when processing each data p in the stream during the detection stage:

1. Update the density of the cell that p belongs to;
2. Update the list of coverage cells, if necessary;
3. Update the representative points, if necessary;
4. Compute IkRD for p.

Please note that Steps 2 and 3 are optional. Execution of Step 2 is dependent on whether or not the coverage requirement is met. If Step 2 is performed, meaning that the list of coverage cells needs to be updated, then Step 3 will be performed as well. If Step 2 is not necessary, then we will see whether p falls into one of the coverage cells. If so, then the representative point extracted from that coverage cell needs to be updated and step 3 will be performed, otherwise Step 3 is unnecessary.

During the detection process, both $k\_dist(c, s)$ and $average\_k\_distance(c_i, s)$ are computed on the fly in SPOT. This enables IkRD to be suitable in applying on data streams.

3.6 Summary

Devising compact data synopsis is one of the central tasks for data stream applications. In this section, we present in detail the data synopsis SPOT uses to capture data information for outlier detection purpose. The data synopses are called Base Cell Summary (BCS) and Projected Cell Summary (the PCS), respectively. BCS captures data information of each populated base cell and can be used to efficiently quantify the PCS thanks to its additive and self-maintainability properties. The PCS contains three data outlier-ness measurements (RD, IRSD and IkRD) and is associated with each non-empty projected cell of the subspaces in SST. One of promising characteristics of the PCS is that it can be incrementally updated as well. This greatly contributes to a good efficiency of SPOT in dealing with data streams.

Chapter 4

SPOT: Stream Projected Outlier Detector

To approach the challenging problem of projected outlier detection in high-dimensional data streams, we propose a new technique, called Stream Projected Outlier Detector (SPOT in short). In this chapter, we will present a detailed discussion of SPOT, with an emphasis on its system architecture, learning stage and detection stage.

4.1 An Overview of SPOT

Our technique for outlier detection in data streams, SPOT, can be broadly divided into two stages: the learning and detection stages. SPOT can further support two types of learning, namely offline learning and online learning. In the offline learning, Sparse Subspace Template (SST) is constructed using either the unlabeled training data (*e.g.*, some available historic data) and/or the labeled outlier examples provided by domain experts. SST is a set of subspaces that features higher data sparsity/outlier-ness than other subspaces. SST consists of three groups of subspaces, *i.e.*, Fixed SST Subspaces ($\mathcal{FS}$), Unsupervised SST Subspaces ($\mathcal{US}$) and Supervised SST Subspaces ($\mathcal{SS}$), where $\mathcal{FS}$ is a compulsory component of SST while $\mathcal{US}$ and $\mathcal{SS}$ are optional components. SST casts light on where projected outliers are likely to be found in the high-dimensional space. SST is mainly constructed in an unsupervised manner where no labeled examples are required. However, it is possible to use the labeled outlier exemplars to further improve SST. As such, SPOT is very flexible and is able to cater for different practical applications that may or may not have available labeled exemplars.

When SST is constructed, SPOT can start to screen projected outliers from constantly arriving data in the detection stage. The incoming data will be first used to update the data summaries (*i.e.*, the PCSs) of the cell it belongs to in each subspace

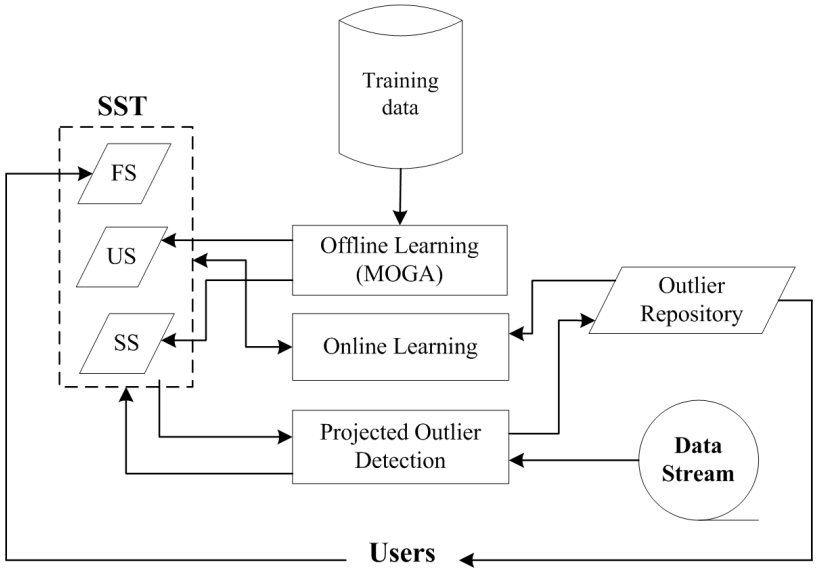

Figure 4.1: An overview of SPOT

of SST. This data will then be labeled as an outlier if the PCS values of the cell where it belongs to are lower than some pre-specified thresholds. The detected outliers are archived in the so-called Outlier Repository. Finally, all or only a specified number of the top outliers in Outlier Repository will be returned to users when the detection stage is finished.

During the detection stage, SPOT can perform online training periodically. The online training involves updating SST with new sparse subspaces SPOT finds based on the current data characteristics and the newly detected outliers. Online training improves SPOT's adaptability to dynamic of data streams.

A system overview of SPOT is presented in Figure 4.1.

4.2 Learning Stage of SPOT

Since the number of subspaces grows exponentially with regard to the dimensionality of data streams, evaluating each streaming data point in each possible subspace becomes prohibitively expensive. As such, we only check each point in a few subspaces in the space lattice alternatively, in an effort to render projected outlier detection problem tractable. In SPOT, we evaluate each data point from the stream in the

subspaces contained in the SST.

We note that detecting outliers is more challenging and difficult than finding all their outlying subspaces. Once an outlier has been flagged in one or more subspaces, it is a fairly trivial task to find its other outlying subspaces by applying some appropriate search method, such as Multiobjective Genetic Algorithm (MOGA). The flagging of outliers is performed online while the search for the outliers' outlying subspaces can be done in an offline manner. Therefore, there is no need to require SST to contain *all* the outlying subspaces for any given projected outlier. *The problem now becomes how to enable SST to contain one or more outlying subspaces for as many projected outliers in the streams as possible.*

In SPOT, SST consists of a few groups of subspaces that are generated by different underlying rationales. Different subspace groups supplement each other towards capturing the right subspaces where projected outliers are hidden. This helps enable SPOT to detect projected outliers more effectively. Specifically, SST contains the following three subspace groups, *Fixed SST Subspaces ($\mathcal{FS}$), Unsupervised SST Subspaces ($\mathcal{US}$)* and *Supervised SST Subspaces ($\mathcal{SS}$)*, respectively. Since the construction of $\mathcal{FS}$ does not require any learning process, the major task of the offline learning stage is to generate $\mathcal{US}$ and/or $\mathcal{SS}$. SST is obtained through the offline learning process using a batch of training data. SPOT is mainly designed as unsupervised outlier detection method (by means of $\mathcal{FS}$ and $\mathcal{US}$). However, a salient feature of SPOT is that it is not only able to deal with unlabeled data but also provides facility for learning from labeled outlier exemplars through $\mathcal{SS}$.

Note that we do not expect that $\mathcal{US}$ and/or $\mathcal{SS}$ cover all the subspaces where outliers may occur. They may be biased by the limited number of top training data or the outlier examples that used for the training purpose. However, the detection of the major portion of outliers are realized by $\mathcal{FS}$. $\mathcal{US}$ and $\mathcal{SS}$ are used only for supplementing $\mathcal{FS}$ and increasing the probability that outliers can be detected.

• **Fixed SST Subspaces ($\mathcal{FS}$)**

Fixed SST Subspaces ($\mathcal{FS}$) contains all the subspaces in the full lattice whose maximum dimension is $MaxDimension$, where $MaxDimension$ is a user-specified

parameter. In other words, $\mathcal{FS}$ contains all the subspaces with dimensions of 1, 2, $\cdots$, $MaxDimension$. $\mathcal{FS}$ satisfies that

$$\forall s, \text{ we have } |s| \leq MaxDimension \text{ if and only if } s \in \mathcal{FS}$$

• **Unsupervised SST Subspaces ($\mathcal{US}$)**

Unsupervised SST Subspaces ($\mathcal{US}$) are constructed through an unsupervised offline learning process. In this offline learning process, SPOT takes in unlabeled training data from the data stream and *automatically* finds the set of subspaces in which a higher number of projected outliers can be detected. Intuitively, these subspaces are where projected outliers are more likely to exist.

We assume that a set of historical data is available for unsupervised learning at the beginning of SPOT. The training dataset should fit into main memory for minimizing possible I/O overhead. Multi-objective Genetic Algorithm (MOGA) is employed to search space lattice to find outlying subspaces of the whole training data for constructing $\mathcal{US}$. To facilitate fitness computation in MOGA, a hypercube H is superimposed in the data space and all the training data are scanned and assigned into one (and only one) cell in H. The BCS of each occupied cell in H are maintained during this data assignment process. When all the training data have been mapped into their corresponding cells, MOGA can be applied on the whole training data to find the subspaces that feature a higher number of outliers. These subspaces will be added to $\mathcal{US}$.

Once we have obtained the initial $\mathcal{US}$, we can further procure more useful subspaces for $\mathcal{US}$. We can find the outlying subspaces of the training data that have the highest overall outlying degree. The selected training data are more likely to be considered as outliers that can be potentially used to detect more subsequent outliers in the stream. The overall outlying degree of the training data is computed in an unsupervised manner by employing clustering analysis.

A key issue in the clustering analysis is to derive the method for accurately measuring the distances between training data points. As the distance between two data points may vary significantly in different subspaces, we therefore expect the distance metric to be able to well reflect the overall outlying (dis)similarity of data points

in difference subspaces, especially those where outliers are likely to be detected. To achieve this, we employ a novel distance metric, called *Outlying Distance(OD)*, for clustering training data. It is defined as the average distance between two points in the top sparse subspaces of the whole training data obtained by MOGA, *i.e.*, $OD(p_1, p_2) = \frac{\sum_{i=1}^{m} dist(p_1,p_2,s_i)}{m}$, where m is the number of top sparse subspaces returned by MOGA and s_i is the i^{th} subspaces in this set.

We utilize *lead clustering method*, also called the fixed-width clustering in [43], to cluster the whole training data into a few clusters. Lead clustering method is a highly efficient clustering method. It adopts an incremental paradigm to cluster data. Each data point p that has yet been clustered in the data set will be assigned to the cluster c' such that $OD(p, c') < d_c$ and $\forall c_i \neq c', OD(p, c') \leq OD(p, c_i)$. The centriod of c' with already m points will be updated upon the cluster assignment of p as $\vec{Cen(c')}_{new} = \frac{m\vec{Cen(c')}+\vec{p}}{m+1}$. If $\forall c_i$, we have $OD(p, c_i) \geq d_c$, then a new cluster is initiated and p becomes the centroid of this new cluster. These steps will be repeated until all the data points in the data set have been clustered.

Due to its incremental nature, lead clustering method features a promising linear scalability with regard to number and dimensions of training data. However, its result is sensitive to the order in which the training data are clustered. To solve this problem, we perform lead clustering in multiple runs under different data orders to diminish its sensitivity to data order. The underlying rationale is that, even though an outlier may be assigned into different clusters in different runs, the chance that it is assigned to a small cluster is relatively high whatsoever. The average size of clusters a point belongs to thus provides an useful insight into its accurate outlying degree, regardless of the data order. The outlying degree of training data p, called *Outlying Factor(OF)*, is defined as $OF^n(p) = \frac{\sum_{i=1}^{n} cluster\_size_i(p)}{n}$, where $cluster\_size_i(p)$ denotes the size of cluster to which p belongs in the i^{th} run and n denotes the number of clustering runs. The sparse subspaces of the top training data, obtained by MOGA, will also be added to $\mathcal{US}$ of SST.

The specification of d_c needs some test and trial work for tuning dc. The alternative parameter of d_c in the lead clustering is the number of cluster to be obtained,

denoted by k. k is obviously easier to be specified than d_c. A few different and reasonable values of k can be used, under which the data are clustered and the Outlying Factor of each training data can be quantified.

In a summary, the subspaces in $\mathcal{US}$ come from two distinct but probably overlapped groups of subspaces, that are, the outlying subspaces of the whole set of training data and those of the top training data that have the highest overall outlying degree. Both groups of subspaces are obtained using MOGA.

• **Supervised SST Subspaces ($\mathcal{SS}$)**

In some applications, a small number of outlier exemplars may be provided by domain experts or are available from some earlier detection process. These outlier exemplars can be considered as carriers of domain knowledge that are potentially useful to improve SST for a better detection effectiveness. MOGA is applied on each of these outlier exemplars to find their top sparse subspaces. There subspaces are called Supervised SST Subspaces ($\mathcal{SS}$). Based on $\mathcal{SS}$, example-based outlier detection [123] can be performed that effectively detects more outliers that are similar to these outlier examples.

Remarks: $\mathcal{FS}, \mathcal{US}$ and $\mathcal{SS}$ differ in their respective role they play in projected outlier detection. $\mathcal{FS}$ is deterministic and it tries to establish an effectiveness *baseline* for projected outlier detection without performing any learning process. Definitely, $\mathcal{FS}$ is able to detect the projected outliers as long as one of their respective outlying subspaces falls into the $MaxDimension$-dimensional space lattice. Due to an exponential growth in size of this space lattice, $MaxDimension$ is normally set quite small, say 3 or 4. As such, the detecting capability of $\mathcal{FS}$ is limited. It cannot detect those projected outliers if the dimensionality of all their outlying subspaces exceed $MaxDimension$. In contrast, $\mathcal{US}$ and $\mathcal{SS}$ are randomized by nature and are not subject to the dimensionality constraint. $\mathcal{US}$ and $\mathcal{SS}$ can help detect more subsequent outliers that share similar characteristics of the top outlying training data or the outliers provided by human users, on which $\mathcal{US}$ and $\mathcal{SS}$ are built. They can supplement $\mathcal{FS}$ to detect projected outliers whose outlying subspace are not located in the $MaxDimension-$dimensional space lattice. **(End of Remarks)**

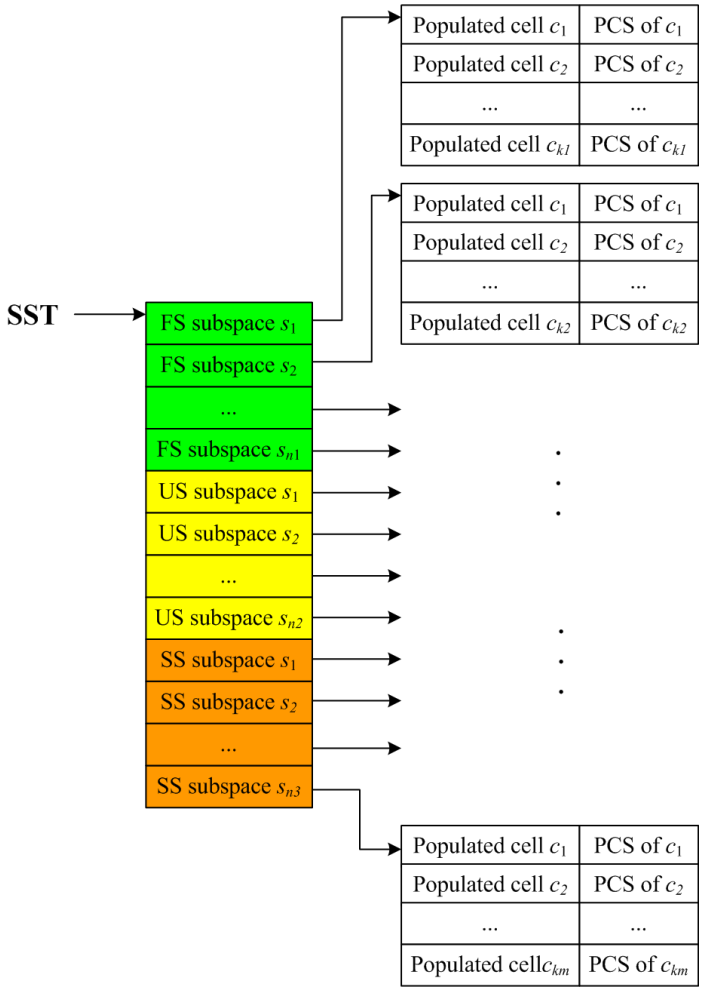

Figure 4.2: The data structure of SST

Based upon the three constituting subspace groups, SST of the whole training data set D_T can be expressed as follows:

$$SST(D_T) = \mathcal{FS} \cup \mathcal{US} \cup \mathcal{SS}$$

where each constituting subspace groups can be expressed as:

$$\mathcal{FS} = \cup_i s_i, \ |s_i| \leq MaxDimension$$
$$\mathcal{US} = TSS(D_t) \cup_j TSS(p_j)$$
$$\mathcal{SS} = \cup_t TSS(o_t)$$

where s_i denotes the i^{th} subspace in the $MaxDimension$-dimensional full lattice, $TSS(D_t)$ denotes the top sparse subspaces of the whole training data D_t, $TSS(p_j)$

Algorithm: SPOT_Unsupervised_Learning ($D_T, d_c, top\_k, N_{runs}$)

Input: Training data D_T, clustering distance threshold d_c, the number of top subspaces or outliers to be chosen top_k, number of clustering runs N_{runs};
Output: Unsupervised SST subspaces;
1. $US \leftarrow$ top_k sparse subspaces from **MOGA**(D_T);
2. FOR i=1 to N_{runs} DO {
3. $D'_T \leftarrow$ **ChangeOrder**(D_T);
4. **Cluster**(D'_T, US, d_c); }
5. $OF\_set \leftarrow$ **ComputeOF**(D_T);
6. $top\_training\_data\_set \leftarrow$ **Select_top_k**(OF_set, D_T);
7. FOR each data p in $top\_training\_data\_set$ DO {
8. $US \leftarrow US \cup$ top_k sparse subspaces of **MOGA**(p); }
9. $SST \leftarrow US$;
10. **Return**(SST);

Figure 4.3: Unsupervised learning algorithm of SPOT

Algorithm: SPOT_Supervised_Learning (OE, top_k)

Input: Set of outlier examplars OE;
Output: Supervised SST subspaces;
1. $SS \leftarrow \emptyset$;
2. FOR each outlier examplar o in OE DO
3. $SS \leftarrow SS \cup$ top_k sparse subspaces of **MOGA**(o);
4. $SST \leftarrow SS$;
5. **Return**(SST);

Figure 4.4: Supervised learning algorithm of SPOT

denotes the top sparse subspaces of the j^{th} top training data that have the highest Outlying Factor and $TSS(o_t)$ denotes the top sparse subspaces of the t^{th} outlier exemplar available.

It is noted that, from the implementation's perspective, SST is more than a set of subspaces. Under each subspace in SST, we also maintain the set of non-empty cells and the the PCS for each of them in this subspace. A hash function is used to achieve a fast retrieval of cells and their the PCS in a given subspace. This hash function maps the interval index vector of the projected cell where a given data belongs to in each subspace of SST into a scalar value. Suppose that each dimension is partitioned

into N_I intervals and let $[a_1, a_2, \ldots, a_{|s|}]$ be the interval index vector of the projected cell where point p falls into in subspace $s \in SST$, then the hash function can be defined as $hash([a_1, a_2, \ldots, a_{|s|}]) = \sum_{i=1}^{|s|} a_i \cdot N_I^{|s|-i}$. This hash function will not result in collision of hash function values but the hash function values in some cases may be rather large.

The non-empty cells (and their the PCS) in each subspace of SST will be updated incrementally in the subsequent detection stage. The data structure of SST is presented in Figure 4.2. Please note that the three subspace groups in SST are highlighted using different colors in the figure, but they are treated equally in practice and are not distinguished with each other once they have been added into SST.

Learning Algorithms

Figure 4.3 is the unsupervised learning algorithm of SPOT. Steps 4-5 perform lead clustering of the training data D_T. The distance metric used in the clustering, Outlying Distance (OD), is based on the top sparse subspaces of the whole training data obtained by MOGA in Step 2 and 3. Step 6 quantifies the Outlying Factor of each training data. Step 7-9 try to find the sparse subspaces for the top training data that have the highest Outlying Factor. MOGA is applied on each of top training data to find their respective top sparse subspaces. $\mathcal{US}$ is added to SST and returned to users in Step 10-11.

The algorithm for the supervised learning, which is simpler than that of the unsupervised learning, is presented in Figure 4.4. Step 2-4 find the sparse subspaces for each outlier exemplar using MOGA. $\mathcal{SS}$ is added to SST and returned to users in in Step 5-6.

4.3 Detection Stage of SPOT

The detection stage performs outlier detection for arriving stream data. As streaming data arrive continuously, the data synopsis the PCS of the projected cell where the streaming data belongs to in each subspace of SST are first updated in order to capture new information of the arrived data. A hash function is employed here to quickly map a data into the cell it is located in any subspace. Then, the data is labeled as a

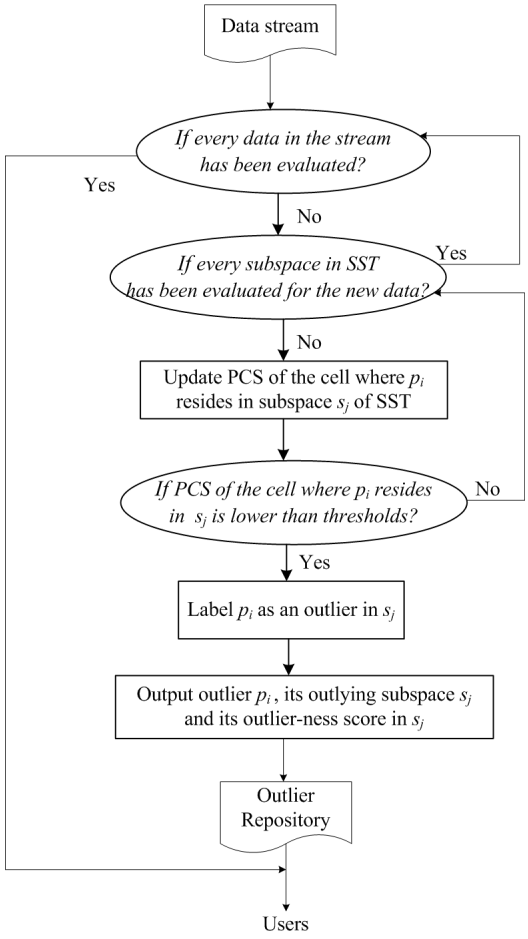

Figure 4.5: The steps of detection stage in SPOT

projected outlier if the PCS of the cell it belongs to in one or more SST subspaces falls under certain pre-specified thresholds. These subspaces are the outlying subspaces of this outlier. All the outliers, together with their outlying subspaces and the PCS of the cell they belong to in these outlying subspaces, are output to the so-called *Outlier Repository*. All or a specified number of the top outliers in Outlier Repository are retuned to the users in the end. The procedure of screening streaming data in the detection stage are illustrated in Figure 4.5.

Due to the speed of data streams and time criticality posed to the detection process, it is crucial that the aforementioned steps can be performed quickly. As we have shown earlier, BCS and the PCS can be updated incrementally and thus will be performed quickly. Also, the outlier-ness evaluation of each data in the stream is

also efficient. It only involves mapping the data point into an appropriate cell and retrieving the PCS of this cell for outlier-ness checking.

An essential issue to the effectiveness of SPOT is how to cope with dynamics of data streams and respond to possible concept drift. Besides using decaying functions on data synopsis, additional efforts have been taken in SPOT to deal with this issue, which are summarized as follows.

- First, both BCSs of non-empty base cells and the PCSs of non-empty projected cells in subspaces of SST will be efficiently maintained as data flow in. This ensures SST to be able to capture the latest data characteristics of the stream and response quickly to data changes;

- Second, any outlier, whenever it is detected, will not be discarded but rather be stored in Outlier Repository. Their top sparse subspaces produced by MOGA will be added into $\mathcal{SS}$ of SST to detect outliers from streaming data arriving later. As a consequence, the detecting ability of SST will be enhanced constantly as an increasing number of outliers are detected along the detection process;

- Third, SST is equipped with an unique ability of *online self-evolution*. The basic idea of self-evolution of SST is that, as the detection stage proceeds, a number of new subspaces are periodically generated online by crossovering and mutating the top subspaces in the current SST. These newly generated subspaces represent the new evolution of SST. However, they cannot be immediately added to SST due to a lack of the PCS information. Under our (ω, ϵ) time model, they have to wait for a time duration (at least one window length ω) to accumulate necessary statistical information. Once the new subspaces join SST, the whole SST, including the new subspaces, will be re-ranked and the new top sparse subspaces will be chosen to create a new SST.

The detection algorithm of SPOT is presented in Figure 4.6. The detection algorithm of SPOT is executed as long as the end of stream has not been reached. The set called *SST_Candidate* is used to store the new subspaces generated periodically from SST to represent its self-evolution. Upon arrival of a new data, three substeps

Algorithm: SPOT_Detection (SST, t, N_{can}, top_k)

Input: SST, self-evolution time period t and number of new subspaces generated N_{can} in self-evolution of SST;
Output: Outlier_Repository where outliers detected are stored.

```
1.  SST_Candidate← ∅;
2.  WHILE NOT (end of stream) DO {
3.  IF a new data p in the stream arrives THEN {
4.    Update_BCS(p);
5.    Update_the PCS(p, SST, SST_Candidate);
6.    IF (Outlier_Detection(p, SST)=True) THEN
7.      Insert(Outlier_Repository, p); }
8.  IF ((Curent_time–Start_time) mod t=0) THEN {
9.    SST← SST_Evolution(SST, SST_Candidate);
10.    SST_Candidate ← Generate_SST_Candidate(SST, N_can);
11.    For each new outliers o added to Outlier_Repository DO{
12.      SST← SST ∪ top sparse subspaces of MOGA(o); } } }
13.   Return(top_k_outliers(Outlier_Repository, top_k));
```

Figure 4.6: Detecting algorithm of SPOT

are performed (Step 3-7), that are, 1) BCS of the base cell in the hypercube to which the point belong are updated; 2) the PCS of the projected cell in each SST subspace and candidate SST subspace to which the point belongs are updated and 3) this point is checked in each of subspaces in SST to decide whether or not it is an projected outlier. If this point is found to be a projected outlier, it will be added to *Outlier_Repository* where all the detected projected outliers are to be stored. Every time when a time period of t is elapsed, the self-evolution is SST is carried out. It creates a new SST by using the current SST and *SST_Candidate* (Step 9). Then, it generates new subspaces for self-evolution in the next cycle. These new subspaces need to stay in *SST_Candidate* for a while to accumulate sufficient data statistics. In Step 10, a number of new candidate SST subspaces are generated. For those newly detected outliers in the latest cycle, MOGA is applied in Step 11 and 12 to find their respective top sparse subspaces in order to update SST. Finally, the top_k projected outliers detected in the detection stage are returned (Step 14).

4.4 Summary

In this section, we present a detailed discussion of SPOT. We starts with an overview of SPOT, followed by a deep look at its training and detection stages. We present the notion of Sparse Subspace Template (SST), the set of subspaces where streaming data will be evaluated. The members of SST are mainly lower-dimensional subspaces and the size of SST is much smaller than the size of the full space lattice. This helps render the projected outlier detection problem tractable. We have also elaborated on the three constituent subspace groups of SST, *i.e.*, Fixed SST Subspaces ($\mathcal{FS}$), Unsupervised SST Subspaces ($\mathcal{US}$) and Supervised SST Subspaces ($\mathcal{SS}$). $\mathcal{FS}$ is generated deterministically while $\mathcal{US}$ and $\mathcal{SS}$ are non-deterministically generated using MOGA. They supplement each other and collectively contribute to the detection effectiveness of SPOT. Once SST has been constructed, detection of outliers can be performed by SPOT. Those data in the streams that have abnormally low the PCS value are labeled as outliers. The top k outliers featuring the highest the PCS can be picked up at the end of detection process. At the end of this section, we have also pinpointed the strategies employed by SPOT to improve its adaptivity for handling the dynamics of data streams in the detection stage.

Chapter 5

Multi-Objective Genetic Algorithm

To generate Unsupervised SST Subspaces ($\mathcal{US}$) and Supervised SST Subspaces ($\mathcal{SS}$), Multi-objective Genetic Algorithm (MOGA) is used as an effective search method for finding top sparse subspaces of the data. In this section, we will first present an overview of evolutionary multiobjective optimization and then elaborate on the ad-hoc design of MOGA in SPOT.

5.1 An Overview of Evolutionary Multiobjective Optimization

In many real-world applications, multiple, often conflicting, objectives arise naturally. Evolutionary algorithms possess some ideal characteristics for dealing with these optimization problems, making them widely used search strategies for multiobjective optimization for a long time. Evolutionary multiobjective optimization has established as a research domain combining the fields of evolutionary computation and classical multiple criteria decision making.

Evolutionary algorithms (EA) [56] originated in late 1950s, and several evolutionary methodologies have been proposed since the 1970s. EAs represents a class of stochastic optimization methods that simulate the process of natural evolution. They imitate the process of organic evolution [65] in an aim to solve parameterized optimization problems.

EAs base their working mechanisms on the fundamental idea of Darwinian evolution, that is, resources are scarce in nature, which leads to a competition amongst the species. In this competition process, all the species will undergo a selection mechanism, in which only the fittest will be able to survive. As such, the fitter individuals stand a higher chance to mate each other, producing even better individuals as a result in the evolution process. Meanwhile, variation occasionally occur through the

means of mutation. This improves the extent of diversity among the species, and contributes to a greater fitness improvement.

The underlying mechanisms of EAs are simple and they have been proven as a general, robust and powerful search mechanism [112]. They are particularly suitable for tackling complicate optimization/search problems that features

1. Multiple conflicting objectives; and
2. Intractably large and highly complex search spaces.

5.1.1 Multiobjective Optimization Problem Formulation

Suppose an optimization problem, with k objectives, are to be solved. Without losing generality, let us assume this optimization problem as a minimum problem where all the k objectives are to be minimized. The solutions to this problem can be formulated as decision vectors that each can be presented as $\vec{x}= \{x_1, x_2, \ldots, x_n\}^T$ in the decision space X. By applying a mapping function $f : X \rightarrow Y$, the decision vector $\vec{x}$ can be transformed to an objective vector $\vec{y}= (y_1, y_2, \ldots, y_k\}^T$ in objective space Y as

$$\vec{y}=\vec{f}\ (\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})\}^T$$

whereby the quality/goodness of this solution can be evaluated. The goal of the optimization problem is to find the decision vector $\vec{x^*}= \{x_1^*, x_2^*, \ldots x_n^*\}^T$ that produces the optimized objective vector $\vec{y^*}$.

Suppose the objective space Y is a real number data space. We discuss the evaluation of solution quality as follows based on the number of objectives k that is to be optimized:

1. When $k = 1$, the problem is reduced to a single-objective optimization problem. The objective space Y can be represented as $Y \subseteq \Re$. In this case, a solution $x_1 \in X$ is considered better than another solution $x_2 \in X$ if $y_1 < y_2$, where $y_1 = f(x_1)$ and $y_2 = f(x_2)$;
2. When $k > 1$, the problem becomes a multiobjective optimization problem. The objective space Y now becomes a multi-dimensional vector, *i.e.*, $Y \subseteq \Re^k$, rather

than a scalar. In such case, the way for evaluating solutions becomes more complicated than the single objective optimization scenario. *Pareto dominance* is a commonly used concept for determining the goodness of solutions in multi-objective optimization problems. Using Pareto dominance, an objective vector y_1 is considered to dominate another objective vector y_2, denoted as $y_1 \succ y_2$, if no component of y_2 is smaller than the corresponding component of y_1 and there are at least one component of y_2 is larger than that of y_1. Consequently, we can say that a solution x_1 is better than (i.e dominate) another solution x_2, denoted as $x_1 \succ x_2$, if $y_1 \succ y_2$.

Based on the concept of Pareto dominance, we say that a solution $\vec{x}^* \in X$ is *Pareto optimal* if for every solution $\vec{x} \in X$, we have

$$\forall i \in I,\ f_i(\vec{x}^*) \leq f_i(\vec{x})$$

and

$$\exists j \in I, f_i(\vec{x}^*) < f_i(\vec{x})$$

where I is the attribute set of the data stream.

In other words, $\vec{x}^*$ is Pareto optimal if there exists no feasible solution $\vec{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. In practice, the Pareto optimum are not a single solution, but rather a set of non-inferior or non-dominated solutions, called the Pareto set.

In objective space, objective vectors can be organized on a number of *trade-off surfaces*. The objective vectors in the same trade-off surface does not dominate each other, while the superiority (or inferiority) of objective vectors in different trade-off surfaces can be decided. For example, those objective vectors in the trade-off surface that are closer to the origin are better than those in the surface that are far from the origin in a minimization problem. The surface where the optimal solutions are located is called the Pareto Front.

5.1.2 Major Components of Evolutionary Algorithms

In a sense, the goal of multiobjective optimization is to find the Pareto set, the set of solutions that form the Pareto Front in the objective space. Unfortunately, obtaining the Pareto set can be computationally expensive and is often infeasible. Hence, the multiobjective optimization methods cannot guarantee the optimal Pareto front. Instead, they only try to find a good approximation that is as close to the true optimal front as possible.

A number of stochastic search strategies such as random search, hill climbing, simulated annealing and evolutionary algorithms (EAs) [71] have been developed to approach optimization problems. Nevertheless, EAs work with an entire population of current solutions rather than a single solution. This is one of the major reasons why EAs are more effective than either hill-climbing, random search or simulated annealing techniques. EAs use the essence of the techniques of all these methods with recombination of multiple solutions in a population. Appropriate operations are defined in order to imitate the recombination and mutation processes as well, and the simulation is complete [14].

An EA typically have following three salient characteristics:

1. A set of solution candidates is maintained;

2. A mating selection process, called selection operator, is performed on this set;

3. Several solutions may be combined in terms of crossover or each solution may be mutated to generate new solutions.

In EAs, the solution candidates are called *individuals* and the set of solution candidates is called a *population*. Each individual represents a possible solution, *i.e.*, a decision vector, in the decision space to the problem under study. Each individual is encoded in EAs based on an appropriate representation such as binary string, tree and so on. The individuals are generated in a generation-wise fashion in the evolution and it is expected that individuals in the newer generations generally outperform their precedents even though it may not the case in practice.

The mating selection process usually consists of two stages: *fitness assignment* and *selection* (in some literatures, selection is called sampling). In the first stage, the individuals in the current population are evaluated in the objective space and assigned a scalar value, *i.e.*, the fitness, reflecting their quality. Then, a mating pool containing the individuals that are used to generate new individuals for the next generation, is created by random sampling from the population according to the fitness values of individuals.

Once the mating pool is generated, it is ready to produce the new individuals of a new generation by employing the variation operators. The variation operators are also commonly referred to as *search operators* as they introduce the mechanism for searching different possible solutions by producing new individuals. The two major variation operators used in EAs are *crossover (recombination) operator* and *mutation operator*. The crossover operator takes a certain number of parents (typically two) each time and creates a child by combining parts of the parents. A crossover probability is usually used in this operator to decide the crossover locus of the parent individuals involved in the crossover for recombination. The mutation operator, in contrast, modifies individuals by changing small parts in the representation according to a given mutation probability. Note that it is possible that, due to randomness of variation, some individuals in the mating pool may not be affected by variation and therefore simply represent a copy of a previously generated solution.

5.1.3 Important Issues of MOEAs

Besides the major constituting components of evolutionary algorithms that have been discussed earlier, there are several important issues needed to be addressed, particularly for multiobjective evolutionary algorithms. These issues include *fitness assignment*, *diversity encouragement* and *elitism*.

It is noted that, from a historical view, these important issues are considered with different extent of attention at the different developmental stages of the field of evolutionary multiobjective optimization. The focus of the evolutionary multiobjective optimization in the first generation lied in approaching the problem of better

approximating the Pareto set; guiding the search towards the Pareto Front [47] [69] [102]. The methods of the second generation incorporated the concept of niching in order to encourage the solution diversity [48] [63] [105]. The importance of elitism was recognized and supported experimentally in the late nineties [98] [119] [130], and most of the third generation MOEAs are equipped with the mechanism of elitism, in one way or another, in order to continuously improve the overall fitness of solutions in the evolution [36] [70] [127].

Fitness Assignment

In the single-objective optimization problem, objective function and fitness function are often identical and are usually used exchangeably. Hence, quality evaluation of individuals in single-objective optimization problems can be conducted directly based on their objective function values. In contrast, the objective function and fitness function differ in evolutionary multiobjective optimization. The objective function f, by applying on the decision vector X, produces a k-dimensional objective vector Y. Quality evaluation of individuals in MOEAs using only Y is not straightforward. Usually, it is desired to convert the k-dimensional objective vector Y to a scalar fitness function in order to facilitate individual quality evaluation. Fitness assignment is the operation to fulfill such conversion. There are three major classes of methods for performing fitness assignment, that are *aggregate sum approaches*, *criterion-based approaches* and *Pareto dominance-based approaches*.

The simplest and most straightforward approach for generating the trade-off surfaces is to aggregate the objectives into a single parameterized objective function. This approach is called *aggregate sum approach*. The parameters of the aggregation function are systematically tuned during the optimization process in order to find a set of non-dominated solutions instead of a single trade-off solution. The most commonly used aggregation function in MOEAs is the weighted-sum aggregation, where the parameters for different objectives are their assigned weights indicating their relative importance in the optimization [62] [64]. Even though it is simple and easy to implement, aggregating function approach suffers major drawbacks that, in most cases, it is not meaningful to directly combine different objectives and, even so, it is

not easy to set the values of the weights used to balance the significance of different objectives in the aggregate function.

Criterion-based approaches potentially choose different objectives in selecting solutions in the same or different generations. When an individual is chosen for producing its offsprings, a different objective may be used to decide which member of the population will be copied into the mating pool. There are different ways for picking objectives for selecting individuals. In [102], each objective are used to create an equal portion of the mating pool. A probabilistic method for choosing objectives can be used; a human-specified probability is assigned to each objective that determines the chance of the objective to be selected [69].

Pareto dominance-based approaches, by its name, quantify the individual's fitness based on the concept of Pareto dominance [52]. Several different methods are used to perform fitness assignment based on the Pareto dominance. Generally speaking, these methods are either based on dominance rank, dominance depth or dominance count. [48] uses the dominance rank, *i.e.*, the number of individuals by which an individual is dominated, to determine the fitness values. Also, as we have mentioned earlier, the solution population is divided into several fronts in the objective space and the depth reflects which front an individual belongs to. [36] and [105] draw on the idea of dominance depth to calculate individual's fitness. [127] and [130] also employ the dominance count, *i.e.*, the number of individuals dominated by a certain individual for fitness calculation. Different Pareto dominance-based strategies can be used to together.

In contrast to aggregation-based methods which calculate an individual's raw fitness value independently of other individuals, the Pareto dominance-based approaches take into account the whole population in fitness calculation. In addition, Pareto dominance-based approaches consider all the objectives simultaneously, rather than a single objective at a time as in the criterion-based approaches.

Diversity Encouragement

Diversity encouragement is a key consideration in MOEAs to ensure a good approximation of the Pareto front in objective space. It is highly desirable that the

solutions obtained can distribute as evenly as possible along the front. This can significantly low the chance of ending up with local optimum.

Most MOEAs try to maintain solution diversity for the Pareto front approximation by utilizing density information in the individual selection process. The general principle is that an individual's probability of being selected is penalized more severely when the density of individuals in its neighborhood is higher. Therefore, the central task in preserving solution diversity is to estimate the local density for each individual. Among the major techniques for estimating local density for individuals in MOEAs are *Kernel function-based methods*, *nearest neighbor-based methods* and *grid-based methods*.

Kernel methods [103] define the neighborhood of a point using Kernel function. For each individual x, the distances from x to all other individuals in the population are calculated. Some sorts of influence function are defined to measure the influence of other individuals on x. In principle, the influence function can be an arbitrary function that uses the distance between two individuals as an argument. The influence function can be a square wave influence function where $f(x, x') = 1$ if $dist(x, x') \leq \sigma$, and $f(x, x') = 0$ otherwise. It can also be a Guassian influence function where $f(x, x') = e^{-\frac{dist(x,x')^2}{2\sigma^2}}$. The estimated density of individual x is defined as th sum of influence function of all the other individuals in the population as

$$density(x) = \sum_{i=1}^{P} f(x, x_i), x \neq x_i$$

Fitness sharing, using Kernel density estimation, is the most popular technique used in the field of MOEAs [48][63] [105].

Nearest neighbor techniques are simple ways for local density estimation [103]. They compute the distance of a given individual to its k^{th} nearest neighbor and use this distance as the means to estimate the density in its neighborhood. Usually, the density estimator is a function of the inverse of this distance [127].

Grid-based methods [103] are another class of commonly used density estimators that use a hypercube to quantify local density for individuals within the objective space. The density of the cells in the hypercube provide a quick guidance as to the

whereabout of dense/sparse regions in the objective space. The density around an individual is simply estimated by the number of individuals in the same cell of the grid [70].

Elitism

Elitism is the effort to address the problem of losing those potentially good solutions during the optimization process because of the randomness in MOEAs. If no special measures are taken, MOEAs cannot guarantee the individuals obtained in a newer generation always outperform those in the older one. Good solutions may not be successfully selected and will therefore consequently disappear forever in the population. Elitism tries to maintain the good solutions found in the whole optimization process and allow them for being selected for producing new individuals. Elitism provides an effective means to prevent deteriorating generations in the evolution.

There are two major ways for implementing elitism in MOEAs. One way is to combine the old population and its offspring, *i.e.*, the mating pool after variation, and to apply a deterministic selection procedure to select from the above combination population to obtain individuals for the new generation, rather than directly replacing the old population by the modified mating pool. Also, the a few best solutions can be directly copied to the next generation. The above two approaches are called non-archive approaches in which no addition storage achieve is needed to keep the good solutions.

Alternatively, a special storage space, called *archive*, can be allocated to keep the promising solutions in the population. The top solutions can be copied in each generation into the archive. The archive can be used just as an external storage separated from the optimization algorithm. The solutions stored in the archive is only for archiving purpose that serves the purpose of facilitating users to find the optimal solution at the end of optimization. However, it can also be integrated into MOEAs in each generation by including archive members in the selection process. In other words, the mating pool consists of individuals not only coming from previous generation but also from the good historical solutions stored in the archive.

5.2 Design of MOEA in SPOT

Genetic algorithms, a particular class of evolutionary algorithms, has been recognized to be well-suited to multi-objective optimization problems. In our work, we employ Multi-Objective Genetic Algorithm (MOGA) to search for subspaces where RD, IRSD and IkRD can be minimized for constructing SST. In this subsection, we will dwell on the important design issues of MOGA in SPOT, including fitness function, individual representation, selection operator, search operator, diversity preservation and elitism.

There are four major parameters that are used in the MOGA, *i.e.*, the total number of generations that are to be performed in the MOGA (denoted by N_g), the number of individuals in the population of each generation (denoted by N_p), the probability that the crossover between two selected individuals will be performed (denoted by p_c) and finally the probability that each bit of the individual will be mutated (denoted by p_m). The typical parameter setup in SPOT is $N_g = 100, N_p = 50, p_c = 0.7$ and $p_m = 0.1$. One can change the value specification of these parameters in order to obtain different search workloads and search strengths for the MOGA.

5.3 Objective Functions

5.3.1 Definition of Objective Functions

To use MOGA in the training stage, we need to define the objective functions for two types of data.

The first type of data is *a single data point*. This applies to each top ranked training data and each outlier exemplar. We need to find the outlying subspaces for the top ranked training data and outlier exemplars to generate $\mathcal{US}$ and $\mathcal{SS}$. For a single data point, we have $f(p, s) = f(c, s)$, meaning that the objective function of data point p is the data sparsity (measured by RD, IRSD, IkRD) of the cell c where p belongs to.

The second type of data is *the whole training data* D_t. The outlying subspaces of the whole training data are found at the beginning of the learning process for generating $\mathcal{US}$. If the objective function of D_t is defined in the same way as that

of a single data, *e.g.*, the average data sparsity of populated cells, then we will have $f(D_t, s) = 1$ because RD, IRSD and IkRD are all defined as ratios against their average level. Instead, the objective function for D_t is defined as the number (more precisely, the percentage) of data points with low the PCS (that is, the corresponding RD, IRSD and IkRD in the PCS is low with respect to some user-defined thresholds). The objective functions of D_t are presented as follows.

Definition 5.1. RD objective of a training data set: RD objective of a training data set D_t in subspace s, denoted as $f_{RD}(D_t, s)$, measures the overall sparsity of D_t in s based on the relative density of cells occupied by D_t in s. It is defined as the inverse of percentage of data in D_t whose RD is lower than the RD threshold γ_{RD}:

$$f_{RD}(D_t, s) = \left[\frac{|\{p_i : RD(c, s) < \gamma_{RD}, p_i \in c\}|}{|D_t|}\right]^{-1}, c \text{ is occupied by } D_t \tag{5.1}$$

where $|\{p_i : RD(c, s) < \gamma_{RD}, p_i \in c\}|$ denotes the number of data points in the cells whose RD in s is less than the threshold γ_{RD}. $|D_t|$ is the number of data in D_t. The lower the value of $f_{RD}(D, s)$ is, the higher data sparsity of D_t in subspace s will be.

Definition 5.2. IRSD objective of a training data set: IRSD objective of a training data set D_t in subspace s, denoted as $f_{IRSD}(D_t, s)$, measures the overall sparsity based on the standard deviation of the cells occupied by D_t in s. It is defined as the inverse of percentage of data in D_t whose IRSD is lower than the IRSD threshold γ_{IRSD}:

$$f_{IRSD}(D_t, s) = \left[\frac{|\{p_i : IRSD(c, s) < \gamma_{IRSD}, p_i \in c\}|}{|D_t|}\right]^{-1}, c \text{ is occupied by } D_t \tag{5.2}$$

where $|\{p_i : IRSD(c, s) < \gamma_{IRSD}, p_i \in c\}|$ denotes the number of data points in the cells whose IRSD in s is less than the threshold γ_{IRSD}. The lower the value of $f_{IRSD}(D, s)$ is, the higher data sparsity of D_t in subspace s will be.

Definition 5.3. IkRD objective of a training data set: IkRD objective of a training data set D_t in subspace s, denoted as $f_{IkRD}(D_t, s)$, measures the overall sparsity based on k-relative distance of the cells occupied by D_t in s. It is defined as the inverse of percentage of data in D_t whose IkRD is lower than the IkRD threshold

γ_{IkRD}:

$$f_{IkRD}(D_t, s) = \left[\frac{|\{p_i : IkRD(c, s) < \gamma_{IkRD}, p_i \in c\}|}{|D_t|}\right]^{-1}, c \text{ is occupied by } D_t \quad (5.3)$$

where $|\{p_i : IkRD(c, s) < \gamma_{IkRD}, p_i \in c\}|$ denotes the number of data points in the cells whose IkRD in s is less than the threshold γ_{IkRD}. The lower the value of $f_{IkRD}(D, s)$ is, the higher data sparsity of D_t in subspace s will be.

f_{RD} and f_{IRSD} can be computed quite straightforwardly. The cells whose RD and and $IRSD$ are lower than their corresponding thresholds are first identified and the number of data points in these cells, which are readily available, are summed up in order to quantify f_{RD} and f_{IRSD}.

Since the training stage can be performed in an offline fashion, it is thus possible to have more than one scan of the training data in computing f_{IkRD}. The coverage cells are first found, whereby the representative points can be extracted. We then calculate the k-distance for each populated cell and count the number of data in the cells whose IkRD is lower than its threshold. A more efficient approximation approach for calculating f_{IkRD} is to, instead of computing k-distance for all the populated cells, we only compute k-distance for the non-coverage cells in the subspace. The underlying rationale is that, under a given threshold for IkRD, *i.e.*, γ_{IkRD}, which is typically a small real number, the IkRD value of coverage cells will be likely well exceed γ_{IkRD} and the cells whose IkRD fall below γ_{IkRD} quite likely come from non-coverage cells. Because the number of non-coverage cells is relatively small, this strategy can thus contribute to a significant speedup in the computation of f_{IkRD} without seriously compromising the effectiveness.

5.3.2 Penalized Objective Functions

As we have pointed out, the projected outliers are more likely to exist in those lower dimensional subspaces. Hence, penalty terms are incorporated into the optimizing objectives, RD, IRSD and IkRD, in SPOT to reflect this knowledge and make the search more effective. The penalty terms enable the search to be focused more on the low dimensional subspaces and is able to reduce the generation of high dimensional

subspaces that would otherwise have to be evaluated in the genetic algorithm. More specifically, we pick up an *exponential penalty function* w.r.t the dimensionality of subspaces. Exponential penalty functions are able to penalize much more severely the subspaces with higher dimensions but far more leniently when the dimensionality of the subspaces is decreased. This helps encourage the search process to converge to those relatively low dimensional subspaces as evolution proceeds, but at the same time avoid it to prematurely converge to subspaces with too few attributes, say 1 or 2, which will highly overlap $\mathcal{FS}$. To prevent an overlap between $\mathcal{US}$ (or $\mathcal{SS}$) and $\mathcal{FS}$, we penalize the objective functions using a large term ($Large\_value = 1000$) to ensure the dimensionality of the subspaces in $\mathcal{US}$ and/or $\mathcal{SS}$ obtained by MOGA is larger than $MaxDimension$. The penalized objective functions of a subspace s w.r.t a set of data D (D can be a single data or the whole training data) are defined as follows:

$$
\begin{aligned}
f_{RD}(D,s)' &= f_{RD}(D,s)(1+\frac{e^{|s|-c}}{e^{\varphi-c}}), \text{if } |s| > MaxDimension \\
f_{IRSD}(D,s)' &= f_{IRSD}(D,s)(1+\frac{e^{|s|-c}}{e^{\varphi-c}}), \text{if } |s| > MaxDimension \\
f_{IkRD}(D,s)' &= f_{IkRD}(D,s)(1+\frac{e^{|s|-c}}{e^{\varphi-c}}), \text{if } |s| > MaxDimension \\
f_{RD}(D,s)' = f_{IRSD}(D_t,s)' &= f_{IkRD}(D_t,s)' = Large\_value, \text{otherwise}
\end{aligned} \tag{5.4}
$$

Formally, the penalized objective function f' can be presented as follows:

$$f' = \{f'_{RD}, f'_{IRSD}, f'_{IkRD}\}$$

5.3.3 Incorporating More Objectives

Using multiple objectives, instead of a single one, is generally more effective to identify outliers. An appealing feature of SPOT is its potential to incorporate more objective functions thanks to MOGA. These objectives functions, however, should meet the following requirements:

- First and most basically, they should be able to measure the sparsity of data in the cells;

- Second, the values of these measures should be able to be computed efficiently by aggregating BCS of base cells of in the hypercube;
- Finally, these objective functions can be updated efficiently in an incremental manner in order to deal with data streams.

5.4 Selection Operators

In SPOT, *Pareto based selection* is used to select fitter solutions in each step of the evolution. It is a stochastic selection method where the selection probability of a subspace is proportional to the value of its fitness. The fitness for a subspace s, denoted by $fitness(s)$, is converted from its Pareto count (or dominance count) of s in the whole population, which is defined as the number of individuals that are dominated by s in the population, *i.e.*,

$$fitness(s) = |\{s_i : s \succ s_i\}| \tag{5.5}$$

where $|\{s_i : s \succ s_i\}|$ denotes the number of individuals s_i that is dominated by s.

The probability that the individual s is selected from the population, denoted by $Pr(s)$, is calculated as:

$$Pr(s) = \frac{fitness(s)}{\sum_{i=1}^{P} fitness(s_i)} \tag{5.6}$$

where P is the population size. Comparing with the conventional Roulette Wheel Selection method based directly on the fitness of solutions, Pareto-dominance based selection method can lower the selection pressure and increase the chances that the subspaces with low fitness to be selected into next population. This will help promote individual diversity and prevent MOGA from becoming stuck to the local optimum.

5.5 Search Operators

Crossover and mutation are two most commonly used search operators in genetic algorithm. Following Holland's canonical GA specification, the crossover and mutation used in this research work is *single-point crossover* and *bit-wise mutation*. In single-point crossover, a crossover locus on two parent individuals is selected and all the bits

beyond that locus in the strings are swapped between the two parents, producing two new children. The bit-wise mutation involves flipping each bit randomly and leads to generating a new children. In our work, all the new individuals generated by crossover and mutation are of the same length, *i.e.*, φ, as their parent(s), where φ is number of dimensions of the input database. There are two associated probabilities, p_c and p_m, used to determine the frequencies for applying crossover and mutation, respectively. Please note that the application of crossover and mutation is not mutually exclusive in the sense that each selected pair of parents will go through tests of crossover and mutation to decide which search operator(s) is/are to be applied on them. Normally, we have $p_c >> p_m$, meaning that crossover is performed in a much higher frequency than mutation.

In some applications, it is possible to incorporate problem domain knowledge into the search operators. For example, if it is known that a certain feature is important in the outlier detection process, then we can increase the probability that the bit (representing this feature in the individual representation) is mutated from 0 to 1 and decrease the probability that it is mutated from 1 to 0 and vice versa for those unimportant features.

5.6 Individual Representation

A straightforward yet effective representation scheme for subspaces is the standard binary encoding; all individuals are represented by binary strings with fixed and equal length φ, where φ is the number of dimensions of the dataset. Using a binary alphabet $\Sigma = \{0, 1\}$ for gene alleles, each bit in the individual will take on the value of "0" and "1", respectively, indicating whether or not its corresponding attribute is selected ("0" indicates the corresponding condition is absent and vice versa for "1"). For a simple example, the individual "100101" when the dimension of data stream $\varphi = 6$ represents a 3-dimensional subspace containing the 1^{st}, 4^{th} and 6^{th} attributes of the dataset.

However, for high-dimensional data streams with φ dimensions, we will end up using a long binary string with a length of φ for representing each subspace. A high

computational overhead is involved as a result in the frequently preformed crossover and mutation operations based on the long binary strings. Therefore, a short representation of individuals is desirable. To this end, we employ a more compact representation of individuals in the MOGA that features a much shorter length. The basic idea of this compact representation is to use an integer string, instead of a binary string, to represent each subspace. Each integer can represent a few binary bits, which contributes a remarkable reduction of the length of individual representation.

If we assume that the number of binary bits needed for representing each integer is L, $L \leq \varphi$ (integers are in the range of $[0, 2^L - 1]$), then the length of an integer string for representing a subspace will be only approximately $\frac{\varphi}{L}$ of the binary string used to represent the same subspace.

Even though it features a more compact representation and enables faster crossover and mutation operations, the integer string representation is limited in two-fold. In crossover operation, the possible crossover locus will be remarkably reduced due to the shorter length of integer strings. This will greatly limit the capability of exploration of crossover operator. In mutation operation, the probability that one integer a is mutated to another integer b ($a \neq b$) by using integer string representation is equal to a constant of $\frac{p_m}{2^L-1}$, where p_m is probability of performing mutation on the integer level, while such probability is actually variable (under different values of a and b) when using binary string representation. As such, integer string representation changes the behavior of mutation and tends to result in abrupt mutations more frequently, which may lead to loss of useful segments of bits that might just turn out to be part of potentially good solutions.

To solve these problems while, at the same time, preserve the desired advantages of integer representation, we propose efficient methods to seamlessly simulate the genetic operations of binary string using integer string representation. We derive ways for quickly obtaining crossover and mutation results between any pair of integers that are consistent with those of binary strings.

For crossover operation, we propose *Crossover Lookup Table (CLT)*. It is a $2^L \times 2^L$ table with each entry being a pair of integers corresponding to the crossover result of

	0	1	2	3
0	(0,0)	(1,0)	(2,0)	(1,2)
1	(0,1)	(1,1)	(0,3)	(1,3)
2	(2,0)	(3,0)	(2,2)	(3,2)
3	(2,1)	(3,1)	(2,3)	(3,3)

Table 5.1: Crossover Lookup Table ($identifier = 1$, $L = 2$)

a given pair of integers. The crossover locus l_c is generated randomly in the range of $[1, \varphi - 1]$. Notice that, from an integer string's perspective, the crossover locus will only be located in the boundary of two adjacent integers or within a single integer. In the former case, two integer strings can be directly crossovered in the same way as the binary strings. In the later case, all the integers after the one that the crossover locus is located can also be crossovered in the same way as the binary strings. The crossover result of the pair of integers where crossover locus is located can be obtained by looking up the *appropriate* Crossover lookup Table based on the value of (l_c *mod* L). As there are $L - 1$ different crossover locus inside an integer, thus we need to pre-computed $L - 1$ different CLTs. Each table is uniquely identified by an integer $i \in [1, L - 1]$. Table 5.1 gives an example CLT with identifier=1 when $L = 2$. Figure 5.1 is a crossover example of two integers by means of the CLT given in Table 5.1. In this example, $l_c = 3$, meaning that the crossover locus will be within the 2^{nd} integer. An appropriate CLT is looked up (The CLT with $identifier = 1, L = 2$, as shown in Table 5.1, is used) and crossover result for the integer pair (1,2), *i.e.*, (0,3), will be found from the table for performing the crossover for the 2^{nd} integer pair.

For mutation operation, we quantify the *Mutation Transition Probability Table (MTPT)*. It is also a $2^L \times 2^L$ table with each entry representing the mutation transition probability from one integer to another. An integer is mutated to another based upon their transition probability initiated from this integer. Let us suppose that two integers a and b differ in l bits in their binary representations ($0 \leq l \leq L \leq \varphi$), the mutation transition probability from a to b, denoted as $Pr(a, b)$, is computed as $Pr(a, b) = p_m^l(1 - p_m)^{(L-l)}$. Unlike CLT, there is only one possible MTPT that needs to be pre-computed for a given φ and L. A simple example is given here to illustrates

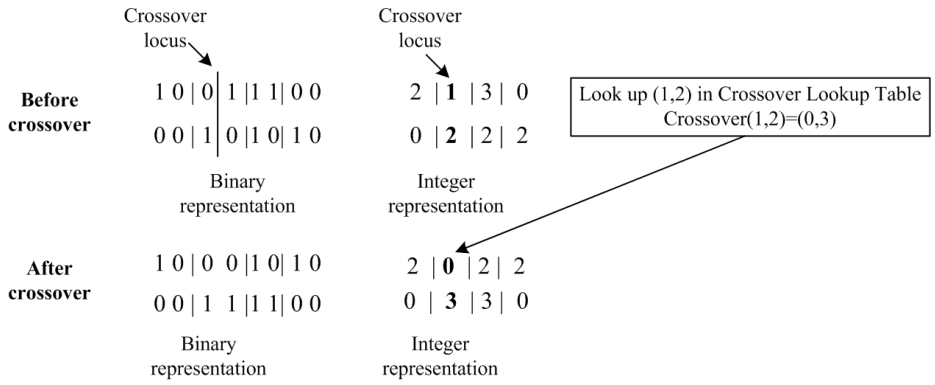

Figure 5.1: An crossover example of two integer strings (with $\varphi = 8$, $L = 2$, $l_c = 3$)

how the mutation transition probability of two integers is computed. Suppose $L = 2$ and $p_m = 0.1$, we need to compute the mutation transition probability from integer 2 to 3. As they differ in only 1 bit in their binary representations, thus $Pr(2, 3)$ can be computed as $Pr(2, 3) = 0.1^1 \cdot (1 - 0.1)^{2-1} = 0.09$.

In the possible case that φ is not dividable by L (that is $mod(\varphi, L) \neq 0$), special treatment will be given for crossover and mutation of the *last pair of integer* in the string. There is because that there are less than L binary bits left for representing this last integer. In crossover operation, if the crossover locus is located within the last integer, we will implicitly image another ($L - \varphi \ mod \ L$) bits of 0 are added to the leftmost of binary strings of the last integer and the Crossover Lookup Table with the identifier of ($l_c \ mod \ L + L$ - $\varphi \ mod \ L$) is looked up for the crossover result of the last pair of integers in the string. For mutation operation, the last integer is subject to the constraint that it can only be mutated to integers in the range of $[0, 2^{L'}]$, where L' is the number of bits representing the last integer and $L' = \varphi \ mod \ L$, whose transition probabilities can be easily derived by the existing Mutation Transition Probability Table.

When using a shorter integer representation for subspaces in MOGA, we can achieve an approximate $\frac{\varphi}{L}$ times performance boost in genetic operations for φ-dimensional streaming data if each integer in the string is represented by L binary bits. The pre-computed CLT and MTPT contribute to a remarkable performance gain of crossover and mutation in MOGA by transforming them to simple lookup

operations from lookup tables, without the frequent on-the-fly conversion between integer and binary strings.

Dr. John Mchugh suggests another method for performing crossover even without using the lookup table. Please see the footnote below for this method. [1]

5.7 Elitism

As we have mentioned earlier, when creating a new population by crossover and mutation, we have a high chance to lose the best subspace(s) found in the previous generation due to the nature of randomness of evolution. Hence, besides Pareto-based selection method, we also use *elitism* method to achieve a constantly improving population of subspaces. We adopt the archive fashion to implement elitism in SPOT.

In our elitism strategy, the best or a few best subspaces are directly copied to the new population, together with other new individuals generated from the mating pool. In MOGA, the non-dominated solutions are deemed as best solutions. The rest of the population is constructed from dominated solutions by Pareto-based selection method if the number of non-dominated solutions has not exceeded the population size P.

In addition, all the solutions obtained in GA in different generations will be stored in an external archive. In the current implementation of SPOT, there is no limit on the archive that is used to store all the solutions that are obtained in the GA. The reason is that the space overhead for storing all the solutions is still tolerable. For example, the size of the archive will only be around 600k if there are 100 generations to be performed in the GA and the population size in each generation is 50.

Elitism is advantageous as it can rapidly increase the performance of MOGA by

[1] let L' be the number of dimensions. let L be L' "rounded up" to a multiple of the word length W so that candidates r and s are zero filled to the left to occupy 2L bit strings represented in arrays A and B of size L/W. & is bitwise and, | is bitwise or, ! is bitwise complement.
Step 1. Pick a crossover point C in 1..L';
Step 2. If ((x = C rem W) != 0) {
i = C / W //the index of the crossover word in A, and B create a mask with x bits set to 1 in the low order portion;
a = A(i) & mask; b = B(i) & mask; a = a xor b; b = a xor b; a = a xor b //swap low order bits
mask = ! mask // complement mask so high bits are now 1s
A(i) = A(i) & mask; B(i) = B(i) & mask; A(i) = A(i) | a; B(i) = B(i) | b;
C = C-1 // first whole word to swap; }
Step 3. for (i=C; i¿=0; i–) { A(i) = A(i) xor B(i); B(i) = A(i) xor B(i); A(i) = A(i) xor B(i); }

preventing a loss of the best found solution(s).

5.8 Diversity Preservation

In SPOT, grid-based method is employed to preserve the diversity of individuals in the population. Compared to the Kernel function-based method and nearest neighbor-based method, the grid-based method does not need to compute the pari-wise distance between all the individuals in the population and its complexity is in the linear order of the number of individuals in the population. Consequently, the grid-based method is more efficient than the other two methods whose complexity in local density estimation are in the quadratic order of the number of individuals.

It is worthwhile pointing out that the hypercube used here in preserving diversity of individuals is not the same hypercube used to process the original data. The hypercube used here is generated in the objective space for performing local density estimation of objective vectors, while the hypercube for processing the original data is created in the decision space for facilitating the calculation of objective vectors. To distinguish the hypercube for processing the original data, which is denoted by H, we denote the hypercube for local density estimation as H' here.

As we are optimizing three objectives in SPOT, we need to create a 3-dimensional hypercube H' in the objective space. Each 3-dimensional penalized objective function vector f' will be assigned to an appropriate cell in H'. The density of each cell in H', denoted by $density(c)$, is quantified as

$$f^{niche} = f' \cdot (1 + \frac{density(c)}{P}),\ f' \in c, c \in H' \tag{5.7}$$

In Eq. (5.7), a penalizing term, $\frac{density(c)}{P}$, is further added to the already penalized objective function f' to incorporate niche mechanism.

As the penalized objective function f' is by nature a vector, Eq. (5.7) is equivalent to the following equations:

$$f_{RD}^{niche}(D_t, s) = f_{RD}(D_t, s)'(1 + \frac{density(c)}{P})$$

$$f_{IRSD}^{niche}(D_t, s) = f_{IRSD}(D_t, s)'(1 + \frac{density(c)}{P})$$

Algorithm: MOGA(D, N_s, PS)

Input: Data set D, number of top sparse subspaces to be obtained N_s and parameter set PS of genetic algorithm, including population size P, probability of crossover p_c, probability of mutation p_m.
Output: Top N_s sparse subspaces of D.
1. $Solution\_Set \leftarrow \emptyset$;
2. $S_{pop} \leftarrow$ initial population of P subspaces;
3. WHILE (evolution_stop_criterion=False) DO {
4. FOR each individual s in S_{pop} DO {
5. **Comp_RD**(D, s); **Comp_IRSD**(D, s); **Comp_IkRD**(D, s) }
6. $Solution\_set \leftarrow Solution\_set \cup$ **Non_dominated_solutions**(S_{pop});
7. **Comp_ParetoFitness**(S_{pop});
8. $S_{pop} \leftarrow$ **Selection**$(S_{pop}, Solution\_set)$;
9. $S_{pop} \leftarrow$ **Crossover**(S_{pop}, p_c);
10. $S_{pop} \leftarrow$ **Mutation**(S_{pop}, p_m); }
11. Return top N_s individuals in $Solution\_Set$;

Figure 5.2: Algorithm of MOGA

$$f^{niche}_{IkRD}(D_t, s) = f_{IkRD}(D_t, s)'(1 + \frac{density(c)}{P})$$

$$\text{where } \{f_{RD}(D_t, s)', f_{IRSD}(D_t, s)', f_{IkRD}(D_t, s)'\} \in c, c \in H' \tag{5.8}$$

MOGA will search in the space lattice for those subspaces with optimized f^{niche}_{RD}, f^{niche}_{IRSD} and f^{niche}_{IkRD} values.

5.9 Algorithm of MOGA

The algorithm of MOGA is presented in Figure 5.2. The major differences between MOGA and the single-objective GA lie from Step 5 to 8. In Step 5, the penalized niching functions for RD, IRSD and IkRD are computed for each subspace in the population of the current generation. The best (*i.e.*, non-dominated) solutions in the current generation is stored in the set of *Solution_set* in Step 6. The Pareto-rank based fitness is computed for all the solutions in the population in Step 7. In Step 8-10, the population for the next generation is produced by crossover and mutation of a selected set of good solutions in the current generation.

5.10 Summary

Multiobjective Evolutionary Algorithms (MOEAs) are techniques well suited for solving multi-objective search/optimization problems. Given the multiobjective nature of the problem of finding outlying subspaces in constructing $\mathcal{US}$ and $\mathcal{SS}$, Multiobjective Genetic Algorithm (MOGA), a subclass of MOEAs, is utilized in SPOT. This section first presents a review of MOEAs and then a discussion of an ad-hoc design of MOGA in SPOT for outlying subspace search. A number of important issues concerning MOGA have been addressed including objective function specification, choice of search operator and selection operator, and the strategies used for elitism and individual diversity improvement.

Chapter 6

Performance Evaluation

6.1 Data Preparation and Interface Development

Before presenting the experimental results of SPOT, we need to first conduct data preparation and interface development. We note that the interface mentioned here is not limited to the Graphical User Interface (GUI) through which users can interact with SPOT. In a broad sense, it can be referred to as the mechanisms that enable SPOT to deal with particular applications by bridging the gap between the functionality provided by SPOT (*i.e.*, detecting projected outliers for high-dimensional data streams) and the application-specific goals that we want to achieve. This work is important to ensure a smooth deployment of SPOT in various practical applications.

6.1.1 Synthetic Data Sets

Synthetic data sets are used in the performance evaluation of SPOT. They are generated by two high-dimensional data generators. The first generator SD1 is able to produce data sets that exhibit remarkably different data characteristics in projections of different subsets of features. The number, location, size and distribution of the data are generated randomly. They are thus quite close to real-life data sets. This generator has been used in [131][126][128] to generate high dimensional data sets for outlying subspace detection. The second synthetic data generator SD2 is specially designed for comparative study of SPOT and the existing methods. In SD2, we need to know *a priori* the true projected outliers existing in the data set for effectiveness evaluation. To this end, we conduct the following setup. Two data ranges are defined in SD2 as $R_1 = (a, b)$ and $R_2 = (c, d)$, where $b + l < c$, l is the length of a partitioning interval in each dimension. This ensures that the data points in R_1 and R_2 will not fall into the same interval for each dimension. In SD2, we first generate a large set

of normal data points D, each of which will fall into R_1 in $\varphi - 1$ dimensions and into R_2 in only one dimension. We then generate a small set of projected outliers O. Each projected outlier will be placed into R_2 for all the φ dimensions. An important characteristic of SD2 is that the projected outliers appear perfectly normal in all 1-dimensional subspaces as all of them are masked by normal data in R_2 for all the 1-dimensional subspaces. Finally, to tailor it to fit data stream applications, the time dimension needs to be added into SD1 and SD2. The outliers generated in SD2 are assumed to arrive at $|O|$ randomly distributed time points $t_i \in \{t_1, \cdots, t_{|O|}\}$ of the whole data stream.

6.1.2 Real-life Data Sets

We also use 4 real-life multi- and high-dimensional datasets in our experiments. The first two data sets come from the UCI machine learning repository. They are called *Letter Image* (RD1, 17-dimensions) and *Musk* (RD2, 168-dimensions), respectively. The third real-life data stream is the MIT wireless LAN (WLAN) data stream (RD3, 18 dimensions)[20]. The wireless network logs correspond to the trace data collected from a real wireless LAN (WLAN) over a period of several weeks by MIT (The data set can be downloaded from *http://nms.lcs.mit.edu/~mbalazin/wireless/*). The fourth real-life data set is the KDD-CUP'99 Network Intrusion Detection stream data set (RD4, 42 dimensions). RD4 has been used to evaluate the clustering quality for several stream clustering algorithms [8][9]. This data set corresponds to the important problem of automatic and real-time detection of cyber attacks and consists of a series of TCP connection records from two weeks of LAN network traffic managed by MIT Lincoln Labs [9]. Each record can either correspond to a normal network connection or an anomaly (falling into one of 4 major categories).

The wireless LAN network and the KDD-CUP'99 network applications are all related to anomaly detection. However, the data stream in the wireless network application is unlabeled, while labeled training and test data are available in the KDD-CUP'99 network anomaly detection applications. Thus, different strategies are used for outlier detection in SPOT. In these two applications, data have been modeled

as high-dimensional vectors where each of them contains a number of varied features to measure the quantitative traffic behaviors. Therefore, SPOT can be easily applied to detect outliers from them.

Before using KDD CUP 99 data in our experiments, we will discuss the major problems that this data set (as well DARPA 98 data from where KDD CUP 99 data was derived) has. As pointed by Brugger (http://www.kddnuggets.com/news/2007/n18 /4i.html), the major problems are 1) there lacks a validation to show that they look like real network traffic, and 2) the data rates are far below what will be experienced in a real medium sized network. However, researchers continue to use these data sets for lack of anything better. It was still useful to evaluate the true positive performance of a network IDS; however, any false positive results are meaningless [22]. By simply considering the attack instances as outliers, the experimental work of SPOT using KDD CUP'99 data set only aims to evaluate how well SPOT is able to detect outlying connection records from the data set and the true positive rate (i.e., detection rate) is considered more important than the false positive rate in our evaluation.

Data Preparation for Wireless Network Data Stream

The wireless LAN network we use consists of over 170 access points spreading over three different physical locations. The WLAN consists of clients connected via access points. SNMP (Simple Network Management Protocol) is used to poll access points every 5 minutes for about four weeks in 2002 (from Saturday, July 20th through Sunday, August 17th, 2002). 5-min intervals are chosen to avoid affecting access point performance. Information is collected about the traffic going through each access point as well as about the list of users associated with each access point. For each user, detailed information is also retrieved on the amount of data (bytes and packets) transferred, the error rates, the latest signal strength, and the latest signal quality, etc. The raw data are summarized into connection oriented records by pairing the MAC address that uniquely identifies the client with the access point to which he/she connected. Each data contains 18 attributes.

Attributes	Description
Site	Building where access point is located
Parent	Name of access point
MAC	MAC address of users
IP	Network address of access point

Table 6.1: List of anonymized attributes

Data Pre-processing

Two aspects of data pre-processing work need to be performed before anomaly detection is conducted using SPOT, namely, *feature selection* and *data partitioning*.

• ***Feature Selection***

Feature selection needs to be first performed to identify those attributes that are suitable for anomaly detection purpose. Amongst the 18 attributes in the original data set, the information regarding the *Site*, *Parent* (*i.e.*, access point), *MAC Address* and *Client IP* for each connection is anonymized for confidential reasons. Table 6.1 presents the list of anonymized attributes in the data. These 4 attributes will not be used for anomaly detection purpose. Also, there are another 3 attributes, *i.e.*, *AID*, *State* and *ClassID*, that are deemed irrelevant to anomaly detection and will not be used either. The remaining 11 attributes are regarded relevant. Table 6.2 shows the list of relevant attributes that will be used for anomaly detection.

After the relevant attributes have been identified, data are projected from the full data space into the subspace delimited by the set of relevant attributes. This will lead to a new data stream on which SPOT will be applied. Projected outliers will be detected form the subspaces of the data space formed by these 11 attributes.

• ***Data Partitioning***

Amongst those 11 relevant attributes, the attributes of *Day* and *Moment* are utilized as summarization attributes, instead of the attributes that are directly used in SPOT for constructing data statistics PCS. Using these two temporal attributes, instead of other attributes, for data partitioning are mainly due to the following reasons:

Attributes	**Description**
Day	Date of the poll.
Moment	Time of the poll.
ShortRet	The total number of 802.11 Short Retries incurred across all packet transmission attempts to this station.
LongRet	The total number of 802.11 Short Retries incurred across all packet transmission attempts to this station.
Quality	A numeric measure in [0,100], average to give the idea of mean quality.
Strength	A numeric value in [0, 100] to indicate signal strength.
SrcPkts	Number of observed packets for which this station was the source.
SrcErrPkts	Number of observed error packets for which this station was the source.
DstPkts	Number of observed packets for which this station was the destination.
DstErrPkts	Number of observed error packets for which this station was the destination.
DstMaxRetryErr	Number of observed max-retry error packets for which this station was the destination.

Table 6.2: List of attributes used in anomaly detection

1. The temporal attributes feature an intuitive concept hierarchy and data can be generalized and specified in a natural and meaningful way;

2. It is reasonable to assume data in different temporal contexts exhibit varied data characteristics and distributions, *e.g.*, the data traffic during weekdays are quite likely to be different from that during weekend;

3. As the connection data are polled in a regular frequency, the distribution of temporal attribute values between the whole and any part of the data stream will be very similar. Hence, it is inappropriate to use these temporal attributes to construct data statistics and detect anomalies.

Similar to the time dimension in an OLAP analysis, *Day* and *Moment* dimensions in the data stream can be discretized into desired time granularities for data analysis purpose. For example, *Day* can be transformed into a categorical value in {Weekday (Monday-Friday), Weekend (Saturday-Sunday)} for studying varying network traffic

All	Daytime	Night
Weekday	Weekday-Daytime	Weekday-Night
Weekend	Weekend-Daytime	Weekend-Night

Table 6.3: Temporal contexts for data partitioning

behaviors during the weekdays and weekends. Similarly, *Moment* can be transformed to a categorical value in {Daytime (8AM-6PM), Night(6PM-8AM)} in order to investigate network traffic behaviors during the daily peak and off-peak time periods. The discretization of *Moment* into Daytime and Night is purely subjective and one may come up with different discretization of the *Moment* dimension.

Based on the varying granularities of *Day* and *Moment*, we can further create finer temporal contexts such as *Weekday-Daytime* and *Weekend-Night* by pairing *Day* and *Moment* dimensions. In total, we create 9 possible temporal contexts, shown in Table 6.3, for constructing data statistics PCS to detect anomalies. These temporal contexts form a three-layer lattice of temporal contexts as follows:

- **Top Level**: All the data in the data stream without considering their *Day* and *Moment* information, denoted by *All*;

- **Median Level**: A single temporal context in either *Day* or *Moment*, denoted as *Weekday, Weekend, Daytime* and *Night*;

- **Bottom Level**: The combined/paired temporal contexts by *Day* and *Moment*, denoted as *Weekday-Daytime, Weekday-Night, Weekend-Daytime* and *Weekend-Night*.

Based on the lattice of temporal contexts, different profiles for characterizing the traffic behaviors will be constructed with respect to different temporal contexts. The anomalies will be detected from the data streams that are relevant to each of these 9 temporal contexts.

Anomaly Validation

Given a lack of labeled test data in this data stream, we need to devise a way for validating the detection results of SPOT and other relevant competitive methods for anomaly detection in data streams. Human examination of all the detected anomalies is infeasible given the potentially large number of subspaces that need to be evaluated. An automated validation method is thus desired.

• ***Outlier-ness Metrics for Validation Method***

We use both distance-based and density-based metrics in the validation method. Distance-based metrics are relatively simple and efficient. Moreover, because projected outliers are mainly detected from low dimensional subspaces, thus the effectiveness of distance-based metrics is largely maintained. Density-based metrics are more effective in measuring data's outlier-ness, though their computational complexity is higher.

Broadly speaking, the outlier-ness metrics that are used in the validation method fall into two major categories, the *global outlier-ness metrics* and the *local outlier-ness metrics*.

Global outlier-ness metrics. The global outlier-ness metrics refer to those metrics that measure the outlier-ness of data points from the prospective of how far away they are from the dense regions (or clusters) of data in the data space. The most intuitive global outlier-ness metric would be the distance between a data point to the centroid of its nearest cluster. This metric can be easily extended to considering the k nearest clusters ($k > 1$) [114]. For an easy referral, we term these metrics as $kNN\_Clusters$ metric. The $Largest\_Cluster$ metric used in [79] is also a global outlier-ness metric. It measures the outlier-ness of a data point using its distance to the centroid of the largest cluster in the data space.

In this study, we use the metric of k-Relative Distance (kRD) in the validation method for identifying anomalies. Given parameter k, kRD of a point p in subspace s is defined as

$$kRD(p, s, k) = \frac{k\_dist(p, s)}{average\_k\_dist(D(p), s)}$$

where $k\_dist(p, s)$ is the sum of distances between p and its k nearest neighbors and $average\_k\_dist(D(p), s)$ is the average of such distance sum for the data points that arrive before p.

Using kRD, we can solve the limitations of $kNN\_Clusters$ and $Largest\_Cluster$ metrics. Specifically, we can achieve a better outlier-ness measurement than $kNN\_Clusters$ metrics when the shapes of clusters are rather irregular. In addition, we can relax the rigid assumption of single mode distribution of normal data and lessen the parameter sensitivity of the Largest_Cluster method. The advantages of using kRD are presented in details as follows:

1. First, multiple cluster representative points are used in calculating kRD. This enables kRD to deliver a more accurate measurement when the cluster shapes are irregular. The representative points, being well scattered within the cluster, are more robust in capturing the overall geometric shape of the clusters such that the distance metric can deliver a more accurate measurement of the distance between a data point and the dense regions of data sets, while using the centroid is incapable of reliably achieving this. For instance, in case of a prolonged shaped cluster as illustrated in Figure 6.1, it is obvious that data o is an outlier while p is a normal data. However, the distance between p and the cluster centroid is even larger than that for o (see Figure 6.1(a)). This will not happen if multiple representative points are used instead though. In Figure 6.1(b), the representative points in this cluster are marked in red color and we can see that the average distance between p and its nearest representative points are less than that of o, which is consistent with human perception;

2. Second, we do not limit the normal data to reside only in the largest cluster as in the Largest_Cluster method. kRD provides flexibility to deal with cases of both single and multiple modes for the normal data distribution.

3. Finally, we employ the concept of distance ratio, instead of the absolute distance value, in kRD to measure the outlier-ness of data points. Thus, kRD waivers the need to normalize data distance for removing the effect of distance scale in

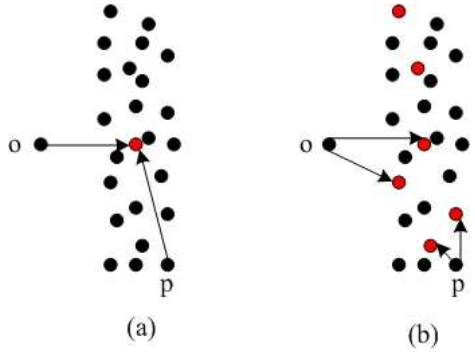

Figure 6.1: Using centroid and representative points to measure outlier-ness of data points

different subspaces. Also, value specification of a threshold parameter based on the distance ratio is more intuitive and easier than the threshold based on the absolute distance.

In the validation method, we do not partition the data space and use the centroid of dense grid cells as representative points for calculating kRD, as we do in SPOT. Instead, we employ the method proposed in [55] to generate representative points for a higher degree of accuracy. The idea of such representative point generation method is that, for a cluster c, a constant number of well scattered points in each cluster are first chosen and these scattered points are then shrunk towards the centroid of the cluster by a fraction of ρ, where $0 < \rho < 1$. Let $rep\_set(c)$ be the set of representative points of cluster c. $rep\_set(c)$ is generated in the following steps:

1. **Initialization step**: As an initialization, the cluster centroid c_0 is inserted into $rep\_set(c)$;

2. **Iterative steps:** For each data p in c that has not yet been included into $rep\_set(c)$, p will be inserted into $rep\_set(c)$ if p is the furtherest data points from all the data currently existing in $rep\_set(c)$. The distance between p and $rep\_set(c)$ is quantitized by the average distance between p and data points in $rep\_set(c)$, *i.e.*,

$$dist(p, rep\_set(c)) = \frac{\sum dist(p, p_i)}{|rep\_set(c)|}, \text{ where } p_i \in rep\_set(c)$$

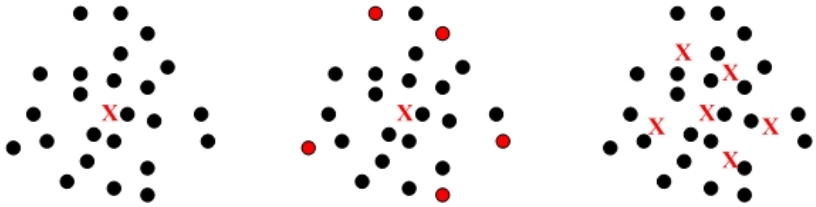

Figure 6.2: Cluster representative points generation

This step will continue until a desired number of scattered points have been obtained in $rep\_set(c)$;

3. **Shrinking step**: Cluster representative points are generated from all the scattered data points in $rep\_set(c)$ by shrinking them towards c_0 by a factor of ρ, *i.e.*,

$$p_i = p_i + \rho(c_0 - p_i), \text{ where } p_i \in rep\_set(c)$$

Figure 6.2 is a step-wise example showing how the cluster representatives are generated. The number of representative points $N_r = 6$ in this example. Figure 6.2(a) is the initialization step where the cluster centroid is selected as a representative point. Figure 6.2(b) shows the well scattered data points selected in the cluster (marked in red color). Figure 6.2(c) shows the shrinking of scattered points obtained in Figure 6.2(b) to eventually generate all the representative points (marked by red crosses).

The steps for computing kRD in the validation method for a data point p in subspace s are as follows:

1. k-means clustering is performed using $D(p)$ in s to generate k clusters;

2. Find N_r representative points from all the clusters using the method proposed in [55]. The exact number of representative points for different clusters can be proportional to the size of clusters;

3. Compute the average distance between each data in the horizon before p and their k respective nearest representative points;

4. Compute the distance between p and its k nearest cluster representative points;

5. Compute kRD as $kRD(p,s,k) = \frac{k\_dist(p,s)}{average\_k\_dist(D(p),s)}$.

Local outlier-ness metrics. Unlike the global outlier-ness metrics, the local outlier-ness metrics typically do not consider the distance between the data to the dense regions of data in the data space. Instead, they quantitize the local data density or distribution in the neighborhood of a data point in order to measure the outlier-ness of this data. As we have discussed in the Chapter of Related Work, the major local distance-based outlier-ness metrics include $DB(k,\lambda)$, $DB(pct,d_{min})$, D^k and D^k Sum. Compared with $DB(k,\lambda)$ and $DB(pct,d_{min})$, D^k and D^k Sum are easier to use. The value specification for k is far more intuitive and straightforward than the local neighborhood parameters used (*i.e.*, λ and d_{min}) in $DB(k,\lambda)$ and $DB(pct,d_{min})$. We will use D^k Sum in the validation method as it is more effective than D^k.

Besides the afore-discussed local distance-based metrics, we also use LOF, a representative local density outlier-ness metrics. Even though there are other existing local density metrics, such as COF and INFLO, that are more effective than LOF in some cases, LOF is works well in practice and it is noticeably faster than COF and INFLO.

• ***Choice of Validation Method***

Given that the data in the wireless network anomaly detection application are unlabeled, we need to choose a method as the validation method for establishing the gold-truth results for performance evaluation. To select a good validation method, we need to investigate the quality of the anomalies detected by different candidate methods. Because the evaluation of anomaly quality is not straightforward, thus we investigate the quality of clusters after the anomalies that are identified by different methods have been removed. Intuitively, the better the clusters are formed after removal of anomalies, the better quality of the anomalies are. In this sense, the overall cluster quality can be used as a useful indicator of quality of anomalies detected by different strategies.

To study the quality of anomalies deleted by different candidate validation methods, we need some objective quality metrics for anomalies. Two major metrics are

commonly used to measure the quality of clusters, *Mean Square Error (MSE)* and *Silhouette Coefficient (SC)* [80].

Mean Square Error (MSE): MSE is a measure for the compactness of clusters. MSE of a cluster c is defined as the intra-cluster sum of distance of c in subspace s, that is the sum of distance between each data in cluster c and its centroid, *i.e.*,

$$MSE(c,s) = \sum dist(p_i, c_0, s), \text{ where } c_0 = \frac{\sum p_i}{|c|}$$

where $|c|$, p_i and c_0 correspond to the number of data within cluster c in s, the i^{th} data within cluster c and the centroid of cluster c in s, respectively.

The overall quality of the whole clustering result C in s is quantitized by the total intra-cluster sum of distance of all the clusters we obtain in s. That is,

$$MSE(C,s) = \sum MSE(c,s), \text{ for } c \in C$$

A lower value of $MSE(C,s)$ indicates a better clustering quality in s. A major problem with MSE is that it is sensitive to the selection of cluster number; MSE will decrease monotonically as the number of clusters increases.

Silhouette Coefficient (SC): SC is another useful measurement used to judge the quality of clusters. The underlying rationale of SC is that, in a good clustering, the intra-cluster data distance should be smaller than the inter-cluster data distance. The calculation of SC of a clustering result C in subspace s takes the following several steps:

1. Calculate the average distance between p and all other data points in its cluster in s, denoted by $a(p,s)$, *i.e.*, $a(p,s) = \sum dist(p, p_i, s)/|c|$, for $c \in C$ and $p_i \in c$;

2. Calculate the average distance between p and data points in each other cluster that does not contain p in s. Find the minimum such distance amongst all these clusters, denoted by $b(p,s)$, *i.e.*, $b(p,s) = min(\sum dist(p, p_i, s)/|c_j|)$, for $c_j \in C$ and $p \notin c_j$ and $p_i \in c_j$;

3. The SC of p in s, denoted by $SC(p,s)$, can be computed by $SC(p,s) = \frac{b(p,s)-a(p,s)}{max(a(p,s)),(b(p,s))}$;

4. The SC of the clustering C in s, denoted by $SC(C, s)$, is simply the sum of SCs for all the data points in the data set, *i.e.*, $SC(C, s) = \sum SC(p_i, s)$, for $p_i \in D$.

The value of SC can vary between -1 and 1. It is desired that SC is positive and and is as close to 1 as possible. The major advantage of SC over MSE is that SC is much less sensitive to the number of clusters than MSE. Given the advantage of SC over MSE, we will use SC as the anomaly quality metric in selecting the validation method.

• ***Quality Evaluation of Anomalies***

In this subsection, we present the quality evaluation of anomalies detected by several different anomaly validation candidate methods by means of SC. We generate four candidates from where the validation method will be chosen. There are the methods use kRD, D^k Sum, LOF, and the method that use all the three metrics. These four anomaly detection methods under study are first applied to detect anomalies. Then, we evaluate the cluster quality using SC. For each candidate method, we select a specific number of top anomalies that have the highest metric values for evaluation. The number of top anomalies we select for each candidate method is identical which is denoted by top_n. It is worthwhile pointing out that the anomalies measured using all the three metrics essentially form a number of trade-off planes where all the anomalies in the same plane are not superior or inferior to each other while superiority or inferiority can be established between anomalies across different planes. The selection process for the top anomalies starts with the best plane until all the top_n anomalies have been selected.

For a fair comparison, we evaluate the cluster quality in the same set of subspaces for all the candidate methods after the detected anomalies have been removed. This set of subspaces can be obtained by a sampling technique. The subspaces are unique and selected randomly from the full space lattice whose maximum dimensionality is bounded by 4. Generating subspaces from this space lattice can ensure that the resulting subspaces are of low dimensionality and contain most of the anomalies that can be detected.

	KDF	D^k_Sum	LOF	$KDF + D^k\_Sum + LOF$
All	0.74	0.62	0.72	**0.75**
Daytime	0.60	**0.77**	0.71	0.67
Night	**0.77**	0.62	0.79	0.72
Weekday	0.71	0.47	**0.80**	0.74
Weekend	0.64	0.66	0.52	**0.69**
Weekday-Daytime	**0.74**	0.70	0.68	0.68
Weekday-Night	0.70	0.76	**0.81**	0.72
Weekend-Daytime	0.73	0.64	0.68	**0.82**
Weekend-Night	0.70	0.74	**0.79**	0.73

Table 6.4: SC results of different validation method candidates

The cluster quality evaluation results are presented in Table 6.4. In this figure, we tabulate the Silhouette Coefficient for all the four candidates of validation methods averaged over the same sampling subspaces. We can see that anomaly detection using LOF and all the outlier-ness metrics perform generally better than the methods that use kDF and D^k Sum; we can achieve better SC in 3 temporal contexts when using LOF alone and the three metrics simultaneously, compared to 2 temporal contexts by using kRD and 1 temporal contexts by using D^k Sum.

This experimental result provides a useful guidance for us to choose a good validation method for different temporal contexts in this study.

• ***Generation of Validation Data Set***

The validation method is only able to deliver good anomaly detection for static databases rather than data streams. In the static databases, one-time execution of the validation method is sufficient for validating all the anomalies in a subspace. Nevertheless, we need to perform the validation method once for each data in a subspace in order to decide whether or not this data is an anomaly in this subspace. This is due to the unique nature of outlier detection in data streams: a data that is labeled as an anomaly at the time of its arrival may no longer be an anomaly later as it may be masked by the data subsequently arrive. As a result, evaluating all the data, even in a single subspace, is extremely expensive. To alleviate this problem, we perform a data sampling to render the validation process tractable.

The validation data set is generated for evaluating SPOT and all the other competitive methods. Different methods under study may produce varied labeling for both the anomalies and normal data for the same data stream. It is desired that the validation data set covers all the anomalies detected by different methods and some, if not all, normal data. The normal data are sampled such that the percentage of anomalies in the whole validation data set is small (say 1%). For SPOT in which multiple outlier-ness metrics are used, the probability of a normal data being selected in the validation is in an inverse proportion to its Pareto ranking if data with stronger outlier-ness are assigned a lower rank. For other competitive methods that use a single outlier-ness metric, such probability is in a right proportion to the normal data's outlier-ness value. By doing so, we can encourage the selection of those normal data with relatively strong outlier-ness (*i.e.*, those data that narrowly pass the outlier-ness thresholds and are therefore labeled as normal) for validation. The rationale for such a skewed selection of normal data is that those normal data that narrowly pass the thresholds are more subject to mis-labeling, compared with those normal data with relatively lower outlier-ness. Thus, the selected set of normal data will be more useful for the validation purpose than a randomly selected set of normal data. The other data that are not either labeled as anomalies by detection methods nor sampled as normal data are assume normal.

Formally, the validation data set, denoted as VD, for a total of m different detection methods, can be expressed as

$$VD = \cup_{i=1}^{m} anomaly\_set(i) \cup_{i=1}^{m} sampled\_normal\_data(i)$$

where $anomaly\_set(i)$ and $sampled\_normal\_data(i)$ represent respectively the anomalies and the sampled normal data labeled by the i^{th} method under study.

This strategy for generating validation data set enables the validation be performed much more efficiently. This is because the number of anomalies detected by different methods are relatively small and the number of normal data to be validated can be well adjusted by users.

• ***Generation of Ground-truth Results***

The validation method is executed on an offline basis to generate the ground-truth results and we do not require this method be able to handle data streams. As a result, more expensive computations, such as multiple scans of data, can be performed to realize an accurate detection of anomalies.

When the validation for a data p in the validation data set VD is performed, we need to specify two configurations:

1. *Subspace configuration*: The subspaces where validation is performed;

2. *Stream length configuration*: The portion of data stream in the horizon W that arrives before p, denoted by $D(p)$.

In other words, the validation of a data point p in validation data set VD is conducted using a subset of data stream, *i.e.*, $D(p)$, in some subspaces. We note that any two validation data points differ in their validation configurations because they will definitely differ in their stream length configuration, regardless of their subspace configuration.

To generate the ground-truth results for validation, we need to, for a validation point p with configuration of s and $D(p)$, quantitize kRD, D^k Sum and LOF for p using $D(p)$ in s. After all the data in VD have been evaluated, data can be ranked. For each data $p \in VD$, the relative rank of p is computed as $relative\_rank(p) = \frac{rank(p)}{|VD|}$. A smaller relative_rank value indicates a higher outlier-ness. p is labeled as an anomaly by the validation method if $relative\_rank(p)$ is smaller than a corresponding threshold, say 0.01. p will be labeled as a normal data otherwise. The data labeling produced by the validation method will be compared with the results obtained by SPOT and other competitive methods for a performance evaluation.

Now, we would like to discuss the approach for ranking validation data. Based on Table 6.4, we can choose the best validation method for different temporal contexts. If a single-metric validation method is used, then each data in VD is simply ranked based on its single-metric value. If multi-metric method is used, all the data in VD are first sorted in a descending order based on their Pareto count, this is, the number

of data in VD that each data dominates. In the case of tie rank of Pareto count, the data involved can be further sorted based on their average rank in each individual criteria (*i.e.*, kRD, D^k, LOF). If both the Pareto count and average ranking cannot break the tie, then the involved data are assigned the same rank.

Data Preparation for KDD-CUP'99 Outlier Detection Data Stream

In many applications, only identifying outliers/anomalies from data streams is not sufficient. Given the inherently varying behaviors of different categories of anomalies, it is desirable that the anomalies we detect could be further categorized into one of known anomaly subtypes for a better understanding of their nature and characteristics. In KDD-CUP'99 outlier detection application, anomalous network connections may be manifestations of outliers that can be divided into as many as 4 different classes. Different classes of attacks may distinguish themselves by anomalous connection behaviors exhibiting in different subspaces. Our task is to, by means of their different connection behaviors in outlying subspaces revealed by SPOT, classify anomalous connection records into one of the known attack classes (or the false-positive class).

•Introduction to KDD-CUP'99 Outlier Detection Application

The KDD-CUP'99 outlier detection data stream contains a wide variety of intrusions simulated in a military network environment. Each instance in this data stream is a vector of extracted feature values from a connection record obtained from the raw network data gathered during the simulated intrusions. A connection is a sequence of TCP packets to and from some IP addresses. The TCP packets were assembled into connection records using the Bro program modified for use with MADAM/ID. Each connection was labeled as either normal or as exactly one specific kind of attack. All labels are assumed to be correct. The attacks will be considered as outliers in our study.

The simulated attacks fall into one of the following four categories: **DoS**: Denial of Service (*e.g.*, a syn flood); **R2L**: unauthorized access from a remote machine (*e.g.*, password guessing); **U2R**: unauthorized access to superuser or root functions (*e.g.*, a buffer overflow attack); **Probing**: surveillance and other probing for vulnerabilities

(*e.g.*, port scanning).

The extract features included the basic features of an individual TCP connection such as its duration, protocol type, number of bytes transferred, and the flag/label indicating the normal or error status of the connection. Other features of an individual connection were obtained by using some domain knowledge and include the number of file creation operations, number of failed login attempts, where root shell was obtained and others. Finally, there were a number of features computed using a two-second time window. These include the number of connections to the same host as the current connection within the past two seconds, percentage of connections that have "SYN" and "REJ" errors, and the number of connections to the same service as the current connection within the past two seconds. In total, there are 42 features for this data stream. In addition, labeled training and test data sets are available in this application. They are the data collected in the first seven weeks of the DARPA experiments.

•***Steps of Applying SPOT on KDD-CUP'99 Outlier Detection Data Stream***

When applying on KDD-CUP'99 Outlier Detection data stream, SPOT takes three major steps in its learning stage to detect outliers, which are presented below:

- **Step 1**: SST is first generated. As a large number of labeled sample outliers are available in this application, SST will contain $\mathcal{SS}$ besides $\mathcal{FS}$ and $\mathcal{US}$. Supervised learning is performed to generate $\mathcal{SS}$ in SST. Since the sample outliers have been assigned varied class labels, we can perform MOGA on all the sample outliers belonging to the same class to produce the $\mathcal{SS}$ for that particular class, and the final $\mathcal{SS}$ in SST contains $\mathcal{SS}$s for the four different classes. That is, $SST = \mathcal{FS} \cup \mathcal{US} \cup \mathcal{SS}\ (OD)$, where OD is whole set of label outlier samples in the data set. $\mathcal{SS}(OD)$ is computed as $\mathcal{SS}\ (OD) = \cup_{i=1}^{4} \mathcal{SS}(OD_i)$, where OD_i is the set containing the outlier samples belongs to the i^{th} attack class.

- **Step 2**: Once we have obtained SST, we need to generate PCS for each subspace in SST to detect anomalies from the data set. Because normal samples are

available in the training data set, thus it is possible for us to use only the normal samples, rather than the whole training data, to construct PCS. This ensures that PCS is constructed in a way to better reflect the normal behavior of data in the application.

- **Step 3**: All the sample outliers in the training data will be evaluated in SST to find their outlying subspaces. Please note that, when we are evaluating each outlier, we only retrieve, but do not update, the PCS of the cell it falls into. This is because the total number of outlier samples is far larger than that of normal samples. Updating PCS using outlier samples will therefore bias it towards outliers, which will disable the ability of SPOT to accurately identify outliers thereafter. When outlying subspaces of all outlier samples are found, signature subspace lookup table will be built. Signature subspace lookup table records the outlying subspaces of anomalies that are used to categorize anomalies. We will discuss it later on in this subsection.

In the detection stage of SPOT, we not only flag out anomalies found in the data stream but also assign labels to them to indicate their membership outlier class. Signature subspace lookup table is used (and updated incrementally) in this stage for anomaly categorization purpose.

To successfully deploy SPOT in this application, we need to carefully address the following several important issues:

- The training data available in this application are not appropriate for the training of SPOT and other anomaly-based detection methods. New training data sets need to be generated to properly support the learning process of SPOT;

- The number of outlying subspaces for even a single outlier returned by SPOT could be large. However, many of these outlying subspaces are redundant and can be removed from the results. We need to develop an algorithm to prune away redundant outlying subspaces to render the result more compact and informative;

- SPOT is only able to identify potential outliers but lacks the facility to correctly categorize them. We need to incorporate categorization functionality into SPOT in this application;

- The problem of high false positive rate is a common yet rarely treated problem for anomaly-based detection methods. It is desired that some mechanisms can be developed to lower the false positive rate and lessen security officer's burden in alarm examination.

• ***Training Data Generation***

The training data set available in this application cannot be directly used by SPOT and other anomaly-based detectors. This is due to the high proportion of attack instances in this training data set; as high as 91% of the samples in this training data are attacks. Normal data behavior is needed in identifying anomalies from the data stream. As such, we need to construct new training data sets based on the original one to meet the distribution assumption that the number of normal connections is much larger than the number of attack connections.

In order to do this, [31] selects from the original training data set all the normal instances and uses sampling technique to sample the attack samples (see Figure 6.3). In this way, a new training data satisfying the distribution assumption is obtained, in which normal connections are dominating the whole data set; as high as 98% of the samples are normal connections while the number of samples for the four attack classes combined amounts to 2%.

However, since only a small number of attack instances are sampled, the new training data set may not be comprehensive enough in terms of capturing sufficient attack instances for generating accurate profiles for different attack classes. Another major limitation of the approach for generating training data in [31] is that the training data set contains outlier samples of different classes. Samples of one class may become noises for another class in the training. To minimize the effect of noises, it would be desired that the training data set contain only outlier samples that belong to the same class. MOGA can be applied to cleaner training data sets to find outlying subspaces that are more relevant to different classes.

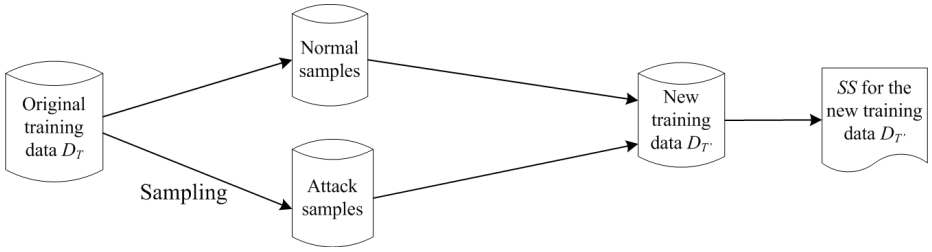

Figure 6.3: Generating single training data set for obtaining $\mathcal{SS}$

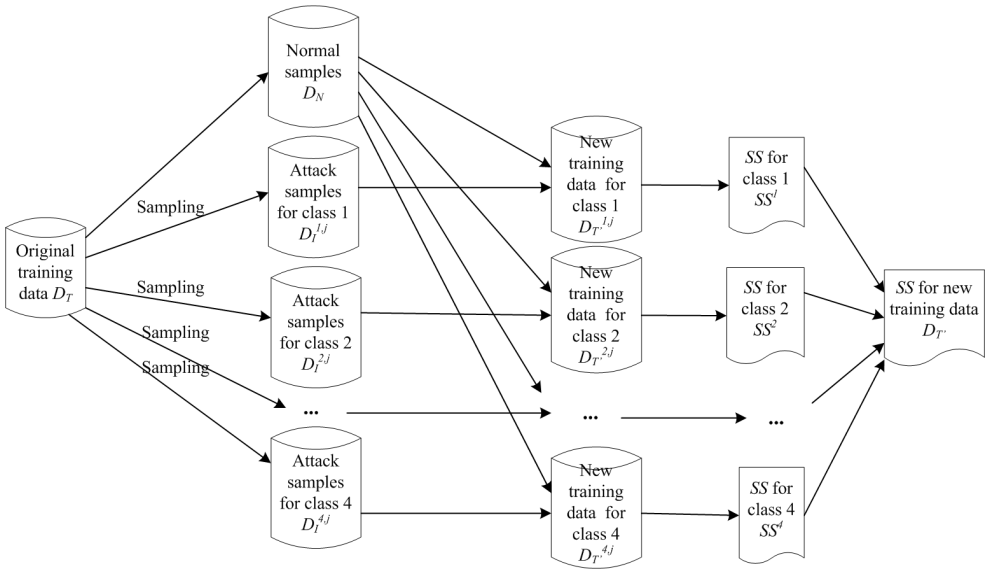

Figure 6.4: Generating multiple training data sets for obtaining $\mathcal{SS}$

To remedy the inherent drawbacks of the single training data set generation method, we adopt a strategy to generate multiple training data sets in order to meet the learning needs of SPOT, as presented in Figure 6.4. The basic idea of this strategy is that, for each outlier class, multiple training data sets are generated and MOGA will be applied on each of them to produce SS for each class. Mathematically, let D_T be the original training data set available. D_T consists two parts, the normal and outlier samples, denoted by D_N and D_I, respectively. D_T can be expressed as

$$D_T = D_N \cup D_I$$

where D_I consists of outlier instances of up to four different classes, so we have

$$D_I = \cup_{i=1}^{4} D_I^i$$

In our work, we generate multiple new training data sets with replacement from D_I^i for each class $i \in [1, 4]$, each such training data set can be expressed as

$$D_{T'}^{i,j} = D_N \cup D_I^{i,j}$$

where j is the number of data sets generated from D_I^i. The data distribution in each new training data set for different classes satisfies the requirement that the normal data dominate the whole training data set. By applying MOGA on $D_{T'}^{i,j}$, we can obtain $\mathcal{SS}$ for class $i \in [1, 4]$, denoted as $\mathcal{SS}^i$, as

$$\mathcal{SS}^{\,i} = \cup_j MOGA(D_{T'}^{i,j})$$

The complete $\mathcal{SS}$ is simply the union of $\mathcal{SS}^i$ for $i \in [1, 4]$ as

$$\mathcal{SS} = \cup_i \, \mathcal{SS}^i$$

By including $\mathcal{FS}$ and $\mathcal{US}$, the complete SST is finally constructed as

$$SST = \mathcal{FS} \cup \mathcal{US} \cup \mathcal{SS}$$

From all the training data sets that have been generated, we can use a small portion of them (say 30%) for cross validation. Cross-validation can help enhance the robustness of SPOT from two aspects. First, it can test whether or not the SST constructed can achieve a satisfactory detection performance (e.g. detection rate $\geq$ 90% and false positve rate $\leq$ 10%) before it is deployed to detect outliers from the data streams. Second, if there are multiple different SSTs generated under different parameter setups, then it is possible to choose a SST that features a higher detection rate and/or a lower false positive rate through cross validation.

•***Redundant Outlying Subspace Removal***

SPOT is able to find outlying subspaces for data in the stream. However, we may obtain a large number of resulting outlying subspaces even for a single data in the stream. Amongst these outlying subspaces, there exists some dominating outlying subspaces that contribute to the outlier-ness of anomalies. Other outlying subspaces can be considered as redundant ones. To facilitate the analysis of anomalies, we

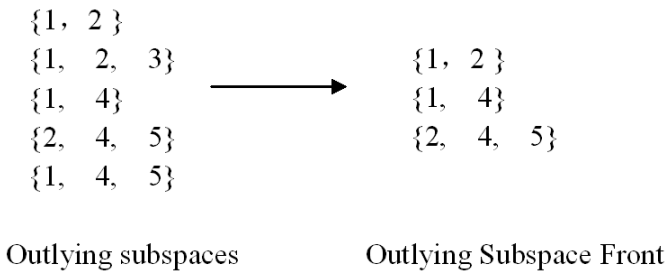

Figure 6.5: Example of outlying subspaces and its corresponding Outlying Subspace Front (OSF) for an anomaly

need to extract the dominating outlying subspaces for anomalies from their outlying subspaces detected by SPOT.

Definition 6.1. Dominating subspace: Let s and s' be two subspaces in the set of outlying subspaces of an anomaly o. If $s \subset s'$, then we call s a dominating subspace over s'. Here $s \subset s'$ (*i.e.*, s is a proper subset of s') if for each dimension $d \in s$, we have $d \in s'$ and $\exists d \in s'$ that $d \notin s$.

In a space lattice where the low order subspaces are positioned in the bottom while the high order subspaces are put on the top, there exists a boundary between dominating outlying subspaces and non-dominating outlying subspace/non-outlying subspaces. In our work, we term this line as *Outlying Subspace Front*. Next, we present the definition of Outlying Subspace Front of an anomaly.

Definition 6.2. Outlying Subspace Front: Let $OS(o)$ denote the set of outlying subspaces of an anomaly o. The Outlying Subspace Front of o is defined as the set of all its dominating subspaces in $OS(o)$, *i.e.*,

$$OSF(o) = \{s, \text{ where } s \text{ is a dominating subspace in } OS(o)\}$$

OSF has the following characteristics:

- A subspace in OSF cannot dominate (or be dominated by) any other subspaces in OSF. They are all partial ordered subspaces;

Algorithm: Find_OSF

Input: $OS(o)$ (Outlying subspace set of o).
Output: $OSF(o)$ (Outlying Subspace Front of o).
1. $OSF(o) \leftarrow \emptyset$;
2. Sort subspaces in $OS(o)$ in an ascending order in terms of their dimensionalities;
3. FOR each existing subspace s in sorted $OS(o)$ DO {
4. $OSF(o) \leftarrow OSF(o) \cup s$;
5. Delete s from $OS(o)$;
6. FOR each existing subspace s' in sorted $OS(o)$ DO
7. IF ($s \subseteq s'$) THEN delete s' from $OS(o)$;}

Figure 6.6: Algorithm for finding Outlying Subspace Front

- OSF is a subset of the corresponding OS and each subspace in OS will be dominated by one or a few subspaces in OSF, *i.e.*, $OSF(o) \subseteq OS(o)$ and $\forall s \in OS(o)$, $\exists s' \in OSF(o)$ that $s' \subseteq s$.

Figure 6.5 presents an example of outlying subspaces of an anomaly and its corresponding Outlying Subspace Front.

The algorithm for finding OSF for an anomaly o is presented in Figure 6.6. The sorting operation in Step 1 of the algorithm helps achieve an early detection of dominating subspaces and, thus, leads to a more efficient pruning of the non-dominated subspaces. Step 2 inserts the next existing subspace in $OS(o)$ into $OSF(o)$. Step 6-7 prune away, for the current subspace s under study in the *For* loop, all the subspaces that are dominated by s. Because of these pruning operations, we can guarantee that the subspaces we insert into $OSF(o)$ are dominating subspaces in $OS(o)$. The whole algorithm is terminated when $OS(o)$ becomes an empty set.

Detection of OSF can have the following advantages:

- The size of OSF for an anomaly is typically far more smaller than the size of its corresponding OS, especially in the case of using a small anomaly-ness threshold. Consequently, storing and processing based on OSF will be much more efficient than the operations based directly on OS;

- OSF is able to capture the subspaces that are truly contributing to the outlierness of anomalies. The existence of a large number of non-dominated (redundant) subspaces may adversely bias the weights of the underlying contributing subspaces in the classification analysis of anomalies.

• ***Anomaly Categorization***

Outlier classification mainly involves categorizing detected anomalies into one of know outlier classes or the class of false positive. We derive categorization functionality and incorporate it into SPOT for achieving this objective.

Signature Subspaces

To achieve anomaly categorization, we generate signature subspaces for each outlier class. The signature subspaces for a target class are those subspaces that can be used to identify outliers for this particular class. To generate signature subspaces for a particular class, we collect the subspaces in OSF of those anomalies falling to this class and use them as the signature subspaces of this class. Mathematically speaking, the set of signature subspaces of class c is defined as

$$Signature(c) = \{s : \exists o \text{ belonging to } c, s \in OSF(o)\}$$

Within a class, different signature subspaces have varying weights to indicate their capability in correctly identifying anomalies for this class. The weighting scheme is necessary in the similarity measure used in the categorization process.

Signature Subspace Weights and Similarity Measure

The weight of a signature subspace s with respect to a class represent the discriminating ability of s towards c. The higher the weight is, the stronger the discriminating power of this subspace is in identifying the instances of class c. Because OSF is by nature a bag of subspaces, we thus borrow the idea of tf-idf (*term frequency-inverse document frequency*) weighting method, a commonly used technique in the domain of information retrieval and text mining, to measure the weight of signature subspaces for each class. The tf-idf weight is a statistical measure used to evaluate how important a term is to a document in a collection or corpus. The importance increases

proportionally to the number of times a term appears in the document but is offset by the frequency of the term in the whole corpus.

Term Frequency (tf). The *term frequency* (tf) for a class is simply the number of times a given signature subspace appears in that class. This count is normalized to give a measure of importance for the signature subspace within the class. The normalization is performed to prevent a bias towards class with larger number of signature subspaces that may feature a higher term frequency regardless of the actual importance of that subspaces in the class. The tf for subspace s_i in class c_j is defined as

$$tf_{i,j} = \frac{N(s_i, c_j)}{N(c_j)}$$

where $N(s_i, c_j)$ denotes the number of occurrences of signature subspace s_i in class c_j and $N(c_j)$ is the number of occurrences of all signature subspaces in class c_j.

Inverse Document Frequency (idf). The *inverse document frequency* (idf) is a measure of the general importance of the term. The idf for signature subspace s_i in class c_j is defined as the inverse of percentage of the classes that contained s_i. Normally, the logarithmic form of this ratio is used for scaling purpose, *i.e.*,

$$idf_{i,j} = log \frac{|C|}{|\{c_j, \text{ where } s_i \in c_j\}|}$$

where $|C|$ corresponds to the total number of classes and $|\{c_j, \text{ where } s_i \in c_j\}|$ is the number of classes that contain s_i.

Finally, the tf-idf weight of signature subspace s_i with regard to class c_j is the product of $tf_{i,j}$ and $idf_{i,j}$, *i.e.*,

$$w_{s_i,c_j} = tfidf_{i,j} = tf_{i,j} \times idf_{i,j}$$

Similarity Measure. Similarity measure needs to be properly defined before the anomalies can be classified. The similarity between an anomaly o and class c is defined as their average inner product, which is the normalized sum of weight products of the outlying subspaces of o and the signature subspaces of class c, *i.e.*,

$$Sim(o, c) = \frac{o \cdot c}{|OSF(o)|}$$

	c_1	c_2	$\cdots$	c_m
s_1	$N(s_1, c_1), T(s_1, c_1)$	$N(s_1, c_2), T(s_1, c_2)$	$\cdots$	$N(s_1, c_m), T(s_1, c_m)$
s_2	$N(s_2, c_1), T(s_2, c_1)$	$N(s_2, c_2), T(s_2, c_2)$	$\cdots$	$N(s_2, c_m), T(s_2, c_m)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
s_n	$N(s_n, c_1), T(s_n, c_1)$	$N(s_n, c_2), T(s_n, c_2)$	$\cdots$	$N(s_n, c_m), T(s_n, c_m)$

Table 6.5: The time-decayed signature subspace lookup table

where $|OS(o)|$ denotes the number of outlying subspaces of o. Let $w_{o,s}$ be the binary vector of o and $w_{c,s}$ be the weight vector of class c. Normally, we assign $w_{o,s} = 0$ if subspace s does not appear in $OS(o)$, so the above similarity measurement can be written as

$$Sim(o, c) = \frac{\sum_{s \in Q} w_{s,o} \cdot w_{s,c}}{|OSF(o)|}$$

Signature Subspace Lookup Table

In order to compute the similarity between anomalies and classes efficiently, we need to have a mechanism to realize fast retrieval of tf-idf information for a give signature subspace. To this end, we construct a signature subspace lookup table to store all the signature subspaces, occurring in different classes, for efficient retrieval. Also, to render it suitable for handling data stream, we incorporate time stamp information to implement time model in this table. We term this table the *time-decayed signature subspace lookup table*, which is defined as follows.

Definition 6.3. Time-decayed signature subspace lookup table. Given the set of classes C and the signature subspaces for all the classes in C, the time-decayed signature subspace lookup table is a $(|S| + 1) \times |C|$ table, where $|S|$ and $|C|$ are the total number of signature subspaces and classes, respectively. C consists of the attack and false-positive classes. The entry of $a_{i,j}, 1 \leq i \leq |s|, 1 \leq j \leq |C|$ is a pair of $< N(s_i, c_j), T(s_i, c_j) >$, corresponding respectively to the count of s_i in c_j and the time stamp when this entry was last updated. The entry of $a_{|s|+1,j}, 1 \leq j \leq |C|$ is also a pair in the format of $< N(c_j), T(c_j) >$, recording the total number of signature subspaces in c_j and the time stamp when this entry was last updated.

An example of the time-decayed signature subspace lookup table is given in Table

6.5. It is worthwhile noting that, in time-decayed signature subspace lookup table, $T(s_i, c_j)$ needs to be updated every time when an anomaly that has outlying subspace s_i is classified into class c_j, and $T(c_j)$ needs to be updated when an anomaly is classified into c_j, regardless of its outlying subspaces.

Based upon the signature subspace lookup table, it will be very efficient to compute td-idf of each signature subspace. First, $tf_{i,j}$ can be computed as follows.

$$tf_{i,j} = \frac{weight(T', T(s_i, c_j)) \cdot N(s_i, c_j)}{weight(T', T(c_j)) \cdot N(c_j)}$$

where T' is the time stamp when the data that has outlying subspace s_i is processed. The information of $N(s_i, c_j)$ and $N(c_j)$ can be directly retrieved for computation from the signature subspace lookup table. The weight coefficients are defined as

$$weight(T', T(s_i, c_j)) = e^{-\frac{\alpha(T' - T(s_i, c_j))}{\Delta t}}$$

$$weight(T', T(c_j)) = e^{-\frac{\alpha(T' - T(c_j))}{\Delta t}}$$

$idf_{i,j}$ can be computed as

$$idf_{i,j} = log \frac{|C|}{|c, \text{ where } s_i \in c \text{ at } T'|}$$

where $|c, \text{ where } s \in c \text{ at } T'|$ denotes the number of classes that contain s at time T'. This only involves counting from the lookup table the classes that contains s_i.

The time-decayed signature subspace lookup table is constructed in the training stage of SPOT using the labeled training data. To do so, we need to register signature subspaces of different classes into this lookup table. Specifically, for each anomaly o found in the labeled training data with its Outlying Subspace Front $OSF(o)$, class label c and time stamp T', we need to register all the subspaces of $OSF(o)$ into the lookup table. This mainly involves initializing and/or updating the counts and time stamps for classes and signature subspaces in the lookup table. Varying updating schemes are used in the following two cases:

1. If $s_i \in OSF(o)$ has already existed in the signature subspace lookup table, then we update class count $N(c)$ and time stamp $T(c)$ of class c as

$$\begin{cases} N(c) = weight(T', T(c)) \cdot N(c) + 1 \\ T(c) = T' \end{cases}$$

and update subspace count $N(s, c)$ and time stamp $T(s, c)$ as

$$\begin{cases} N(s, c) = weight(T', T(s, c)) \cdot N(s, c) + 1 \\ T(s, c) = T' \end{cases}$$

2. If $s \in OSF(o)$ does not exist in the signature subspace lookup table, then we will update class count $N(c)$ and time stamp $T(c)$ of class c as

$$\begin{cases} N(c) = weight(T', T(c)) \cdot N(c) + 1 \\ T(c) = T' \end{cases}$$

and initialize subspace count $N(s, c)$ and time stamp $T(s, c)$ as

$$\begin{cases} N(s, c) = 1 \\ T(s, c) = T' \end{cases}$$

For each class $c' \neq c$, we perform the following initialization:

$$\begin{cases} N(s, c') = 0 \\ T(s, c') = \text{" "} \text{ (null time stamp)} \end{cases}$$

Steps for Anomaly Classification

When constructed, the signature subspace lookup table can be used to classify anomalies in the data stream. Each anomaly is classified into one or more possible attack classes or the false-positive class based on the class membership probabilities of the anomaly. The class membership probability of an anomaly o with respect to class $c_i \in C$ is computed as

$$pr(o, c_i) = \frac{sim(o, c_i)}{\sum_i sim(o, c_i)} \times 100\%, \text{ where } c_i \in C$$

The higher the probability for a class is, the high chance that the anomaly falls into this class.

An anomaly o can be classified into a class c_i if $pr(o, c_i) \geq \tau$, where τ is the similarity threshold. As s result, under a given τ, it is possible that an anomaly o is classified into a few, instead of one, classes if their similarities are high enough. The set of classes that o may fall into, denoted as $class(o)$, is defined as

$$class(o) = \{c_i, \text{ where } pr(o, c_i) \geq \tau, c_i \in C\}$$

For each anomaly, we can further sort its membership classes based on the respective membership probabilities in an descending order. This facilitates users to pick up the top k ($k \in [1, |C|]$) most likely attack class(es) of the anomaly for further investigation.

After the classification of o is finished, we need to update the counts and time stamps for classes and signature subspaces in the lookup table. Such updates reflect the concept drift as the lookup table is updated accordingly in response to the data dynamics by adjusting the weights of signature subspace in the lookup table. A promising characteristic of signature subspace lookup table is that it can be updated incrementally, enabling the update of lookup table to be performed very efficiently in the detection process. For each detected anomaly, the steps of class membership probability computation, class classification and signature subspace lookup table updating are performed in an on-the-fly manner.

•Handle False Positives

False positives, also called false alarms, are those anomalies that are erroneously detected as the attacks by the system. Even though they are benign in nature and not harmful as compared to real attacks, false positives consume a fair amount of human effort spent on investigating them whatsoever and thus making it almost impossible for security officers to really concentrate only on the real attacks. Generally, among all the anomalies detected, up to 90% of them may be false positves. It is much desirable to quickly screen out these false postives in order to allow closer attention to be paid towards the real harmful attacks.

The situation we are facing in the KDD-CUP'99 Outlier Detection application is that there are no available false-positive exemplars in the training data set. Therefore, unlike the attack classes, it is not easy to directly create the signature subspaces for the false-positive class. However, there are a fair amount of normal samples in the training data set. If any of them are found abnormal, *i.e.*, they have some outlying subspaces, then they will be considered as false positives. These false positives from the training data provides the basis for constructing the signature subspace for the false-positive class. Consequently, like other attack classes, the construction of signature subspaces of false-positive class can be started as early as in the learning stage of SPOT.

The set of signature subspaces of the false-positive class starts from an empty set. In the learning stage of SPOT, a normal data in the training data set will be considered as a false positive if it is found abnormal in some subspaces. Formally, a data point p is a false positive if we have

$$OS(p) \neq \emptyset \text{ and } label(p) = normal$$

The moment a data point p is identified as a false positive, the subspaces in its OSF will be properly registered into the current signature subspace lookup table. Ideally, the similarity between false positives and the false-positive class should be significantly higher than their similarities to other attack classes. However, this may not be true in the early stage due to the immatureness of the signature subspaces of the false-positive class. As an increasing number of false positives are continuously registered, the signature subspaces of the false-positive class will keep on growing. As a result, we will witness an continuously improved classification accuracy of false positives. We keep trace of the goodness of the signature subspaces we have constructed thus far at any time of the learning process. As a rule of thumb, the growing stage could be continued until the moment when a satisfied detection accuracy, say 90%, is achieved for false-positive class.

If the training data fail to establish a signature subspace lookup table for achieving a satisfactory classification accuracy, the construction of signature subspaces for the false-positive class will be continued to the detection stage of SPOT. Since the examination of false positives by domain experts during the detection stage is rather

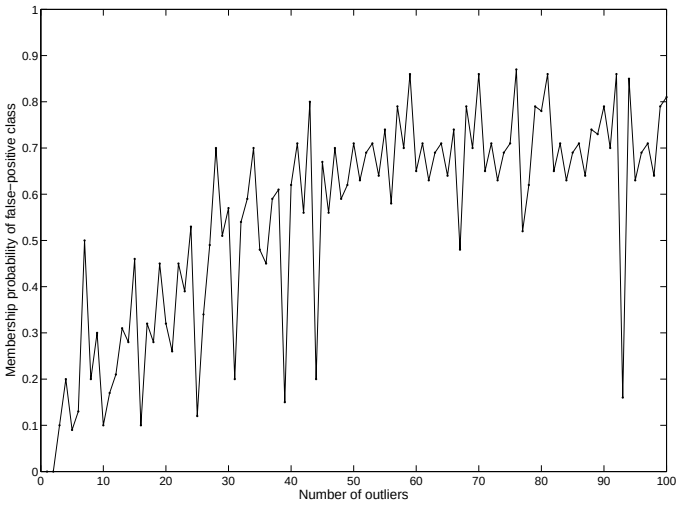

Figure 6.7: Change of the member probabilities of anomalies w.r.t the false-positive class

time-consuming, if it is not completely impossible, we thus employ an alternative automatic approximation method to expand the set of signature subspaces for the false-positive class. The basic idea for this automatic approach is that we collect the anomalies detected by SPOT and label those anomalies as false positives whose probability for falling into any known attack class is lower than a corresponding probability threshold. Given that the overwhelming majority of anomalies detected by SPOT are false positives, it is reasonable to consider all these anomalies that cannot be categorized into any attack classes as false-positives without significantly compromising its detection accuracy. This could save a lot of human efforts taken in anomaly examination.

When labeled samples are absent, we cannot rely on the detection rate to properly pinpoint the transition from the growing stage to the later mature stage. Alternatively, we depend on the changes in the membership probabilities of anomalies with regard to the false-positive class. The higher the membership probability is, the better identification is achieved for the false positives. Such membership probability value is relatively low at the beginning due to the immatureness of the signature subspaces of the false-positive class. When this set grows as time evolves, we gradually obtain

a better detection performance of false positives and the similarity value will be increased. When the similarity value starts to converge (reaching a plateau stage), then we can consider the set of signature subspaces of the false-positive class to be mature. Figure 6.7 shows the change of the membership probabilities of anomalies with respect to the false-positive class. Please note that, after the set of signature subspace has reached the mature stage, we are in a better position to identify anomalies. One such example will be the 93^{th} anomaly shown in Figure 6.7 that has remarkably low probability with respect to false-positive class compared to other anomalies. It is probably an attack instance, though its exact attack class is unknown by solely reading this figure.

6.2 Experimental Results

After data preparation and application interface development have been finished, we can conduct experimental evaluation on SPOT and compare the performance of SPOT with other competitive methods. We use both synthetic and real-life datasets for performance evaluation in our experiments. All the experimental evaluations are carried out on a Pentium 4 PC with 512MB RAM. These experimental results are reported in this subsection.

6.2.1 Scalability Study of SPOT

The scalability study investigates the scalability of SPOT (both learning and detection processes) w.r.t length N and dimensionality φ of data streams. Sine construction of $\mathcal{FS}$ does not require any learning process, thus the learning process we study here refers only to the unsupervised learning that generates $\mathcal{US}$ of SST. Due to their generality and controllability, data sets generated by SD1 with different N and φ are used in all scalability experiments.

Scalability of Learning Process w.r.t N

Figure 6.8 shows the scalability of unsupervised learning process w.r.t number of training data N. The major tasks involved in the unsupervised learning process are

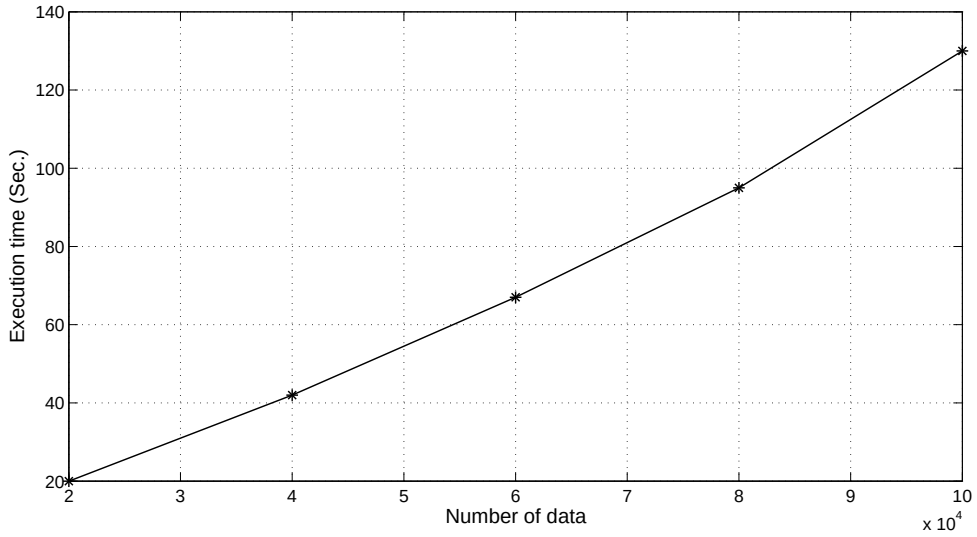

Figure 6.8: Scalability of learning process w.r.t data number

multi-run clustering of training data, selection of top training data that have the highest outlying degree and application of MOGA on each of them to generate $\mathcal{US}$ of SST. The lead clustering method we use requires only a single scan of the training data, and the number of top training data we choose is usually linearly depended on N. Therefore, the overall complexity of unsupervised learning process scales linearly w.r.t N.

Scalability of Learning Process w.r.t φ

Since the construction of $\mathcal{FS}$ in SST does not need any leaning process, we only need to study the effect of data dimensionality on the construction of $\mathcal{US}$. Because $\mathcal{US}$ is only supplemental to $\mathcal{FS}$, a relatively large fixed search workload of the MOGA will be sufficient for most of cases. Under a fixed search workload, the complexity of constructing $\mathcal{US}$ is in an exponential order with regard to the data dimensionality. This is because the computation of the PCSs of projected cells involves data aggregation of the BCSs of their base cells and the number of base cells grows exponentially when the data dimension increases. Figure 6.9 shows the results, which indicate an exponential growth of execution time of learning process when φ is increased from 20

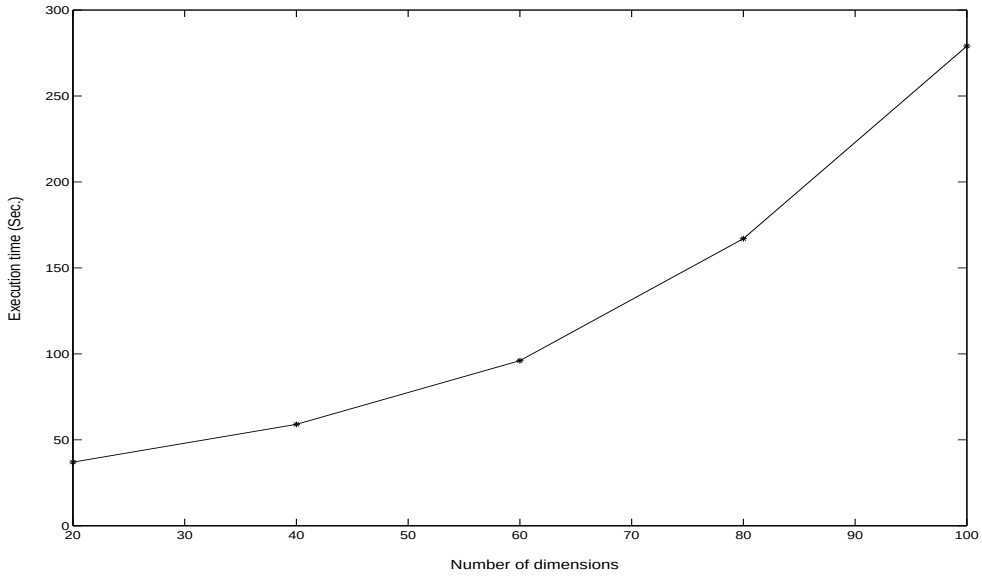

Figure 6.9: Scalability of learning process w.r.t data dimension

to 100 under a fixed search workload in MOGA.

Scalability of Detection Process w.r.t N

In Figure 6.10, we present the scalability result of detection process w.r.t stream length N. In this experiment, the stream length N is set much larger than the number of training data in order to study the performance of SPOT in coping with large data streams. Figure 6.10 shows a promising linear behavior of detection process when handing an increasing amount of streaming data. This is because that the detection process needs only one scan of the arriving streaming data. In addition, BCS and the PCS are both incrementally maintainable, thus the detection process of SPOT is efficient. This leads to a good throughput of SPOT and enables it to deal with fast data streams.

Scalability of Detection Process w.r.t φ

The dimensionality of a data stream φ affects the size of $\mathcal{FS}$ that is used in detection process. When $MaxDimension$ is fixed, the size of $\mathcal{FS}$ is in an exponential order of

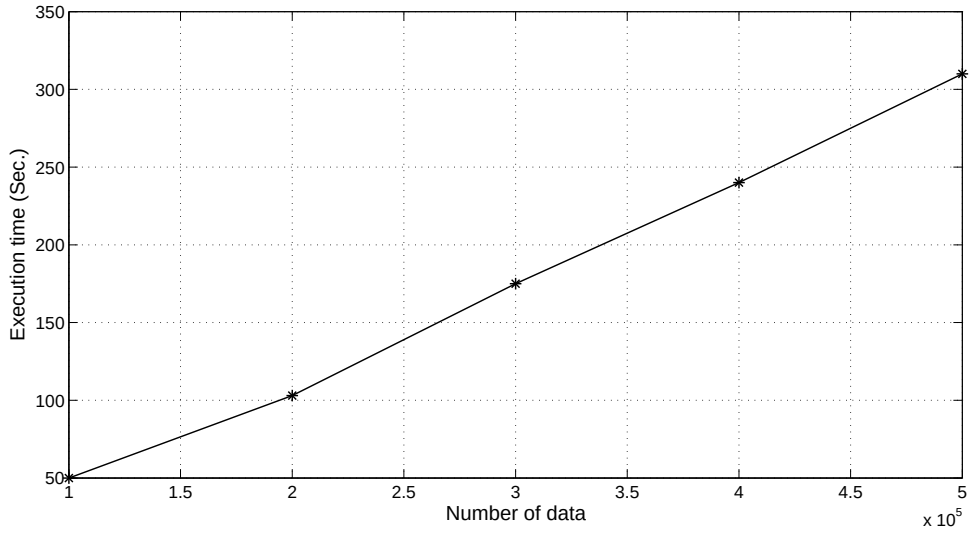

Figure 6.10: Scalability of detection process w.r.t data number

φ, which is usually much larger than that of $\mathcal{US}$ and $\mathcal{SS}$. This causes $\mathcal{FS}$ to dominate the whole SST. As such, the execution time of detection process is expected to grow exponentially w.r.t φ. We typically set lower $MaxDimension$ values for data streams with higher dimensionality to prevent an explosion of $\mathcal{FS}$. In this experiment, we first use $MaxDimension = 3$ for data streams of different dimensions and we can see an exponential behavior of the detection process. Then, we use variable values for $MaxDimension$ to adjust the size of $\mathcal{FS}$. We set $MaxDimension = 4$ for data sets with dimension of 20 and 40, set $MaxDimension = 3$ for data sets with dimension of 60 and finally set $MaxDimension = 2$ for data sets with dimension of 80 and 100. If this strategy is used, we will witness an irregularly shaped, rather than an exponential, curve of the detection process w.r.t φ. The results are presented in Figure 6.11.

Throughput Analysis of SPOT

At the end of scalability study, we would like to carry out the throughput analysis of SPOT. This analysis involves investigating the number of data that SPOT is able to process per second under different number of data dimensions. In this experiment,

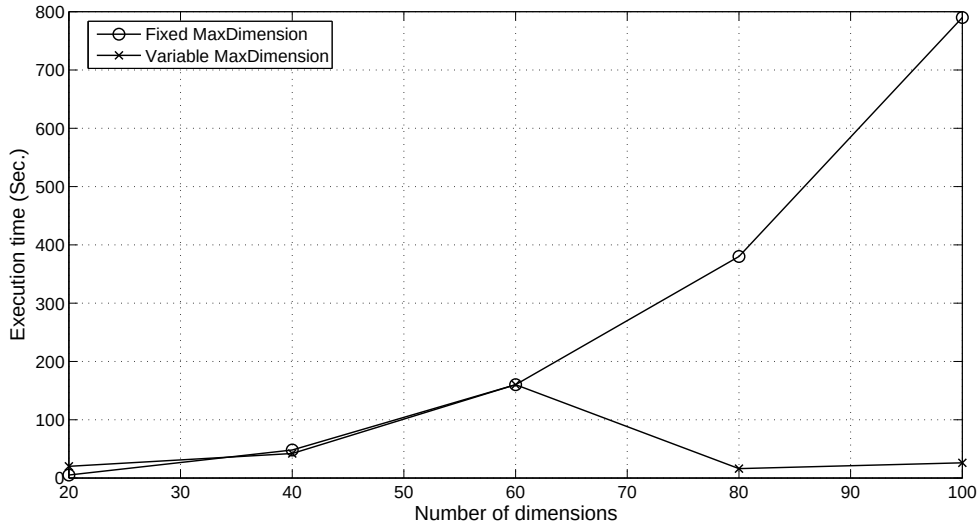

Figure 6.11: Scalability of detection process w.r.t data dimension

we evaluate the throughput of SPOT under five different numbers of dimensions, ranging from 10 to 50. The results are shown in Figure 6.12. Because $\mathcal{FS}$ dominates the whole SST and its size is in an exponential order with respect to the number of data dimensions, thus the time required to process the same amount of data is also approximately in an exponential order of the number of data dimensions. Hence, the throughput of SPOT is decreased exponentially when the number of dimensions increases.

6.2.2 Convergence and Adaptability Study of SPOT

The convergence and adaptability study of SPOT involve the investigation on the convergence of MOGA in SPOT for searching outlying subspaces and the evolution of SST when the outlier detection process evolves in SPOT.

Convergence study of MOGA

We first study the convergence of MOGA in terms of optimizing RD, IRSD and IkRD. Convergence of MOGA is crucial to outlying subspaces searching in SPOT. We investigate the average of RD, IRSD and IkRD of the top 10 subspaces in the population

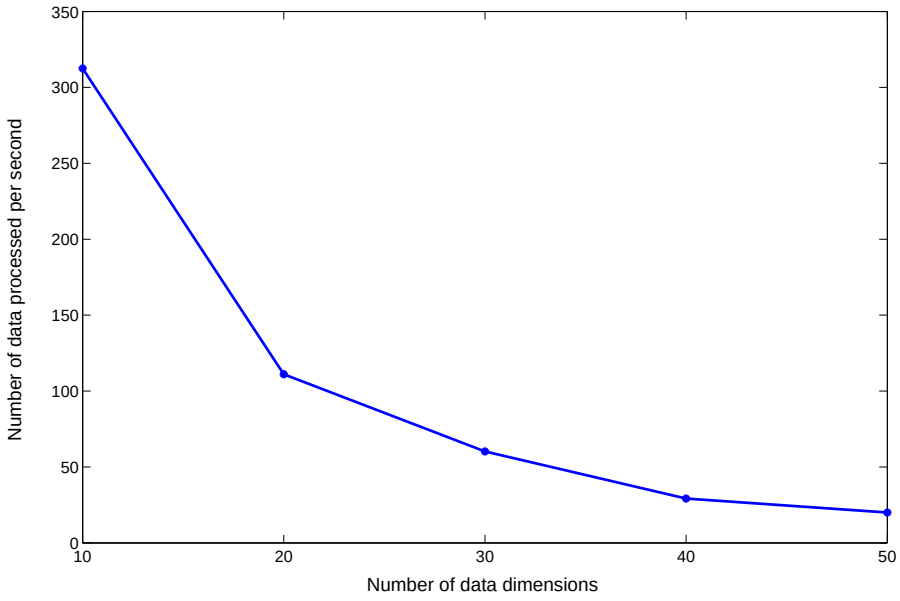

Figure 6.12: Throughput analysis of SPOT

of each generation of MOGA. This experiment is conducted on SD1, RD1, RD2, RD3 and RD4. Only the results of RD4 (KDD-CUP'99 data stream) are presented (see Figure 6.13). Similar results are obtained for other datasets. Generally speaking, the criteria are being improved (minimized) as more generations are performed in MOGA. This indicates a good convergence behavior of MOGA in searching outlying subspaces. However, there are some abrupt increase of optimizing criteria values in the search process. The underlying reason is that, when elitism is not used, there is a higher chance of the occurrence of *degrading generations* in the evolution. Here, degrading generations refer to those generations in the evolution whose fitness function values of a population, especially those of the top individuals, become worse than its previous generation. This is due to the randomized nature of MOGA that likely renders good solutions in one generation to be lost in the next one by crossover or mutation operations.

When elitism is employed, we can achieve a better optimization of RD, IRSD and IkRD. Because the best solutions are copied into the next generation, we do not see any vibrations of optimizing criteria values in MOGA and the optimizing criteria can

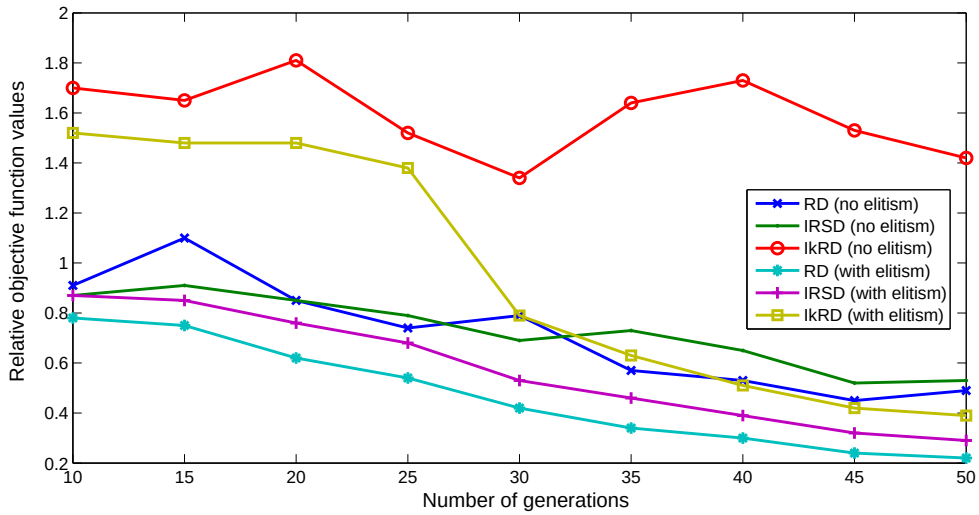

Figure 6.13: Convergence study of MOGA

be constantly improved as the evolution proceeds.

Evolution of SST

One of the important features of SPOT is its capability of self-evolution. This feature enables SPOT to cope with the fast-changing data streams. In this experiment, we investigate the evolution of SST as an informative indicator of concept drift of data streams. The setup is as follows. We keep the initial version of SST (*i.e.*, the SST obtained after the first 1000 data points are processed) and record the versions of SST when every 1000 data (up to 10,000) are processed afterwards. Self-evolution is activated at every 1000-data interval. We compare different SST versions with the initial one and calculate the percentage of identical subspaces. SD1, RD1, RD2, RD3 and RD4 are used in this experiment. The results are shown in Figure 6.14. We find that an increasing number of subspaces in the initial SST version have disappeared in the later SST versions as more data points are processed. We use the same seeds in MOGA, ruling out the randomness in individual generation for different self-evolution sessions. Therefore, the change of SST is due to the changing characteristics of the data stream (reflected by the BCS and the PCS) and outliers we have detected in

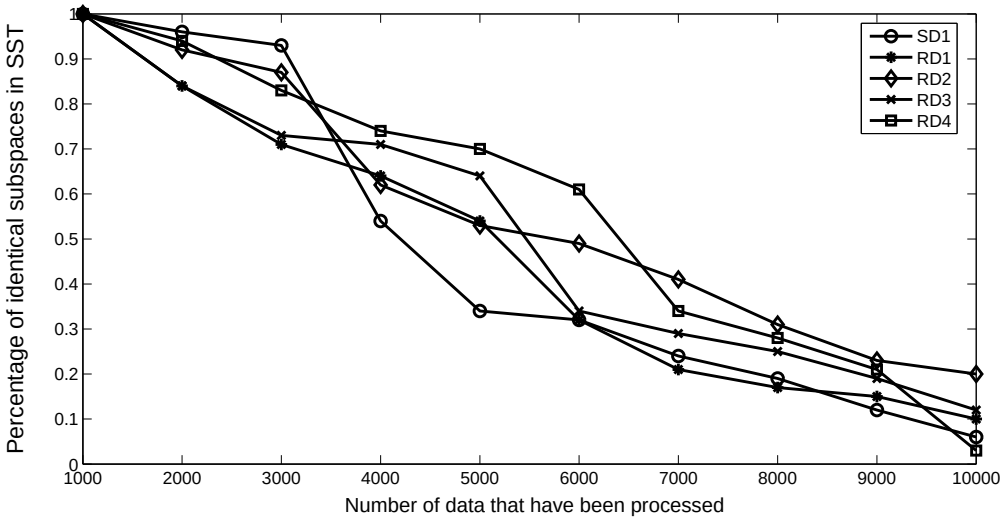

Figure 6.14: Evolution of SST

different stages.

6.2.3 Sensitivity Study of SPOT

In this subsection, experimental study will be conducted to test the sensitivity of SPOT towards the major parameters it uses. We will first investigate of the effect of pre-specified outlier-ness thresholds. Then, we will study the effect of search workload W of MOGA. As multi-run lead clustering is the core technique used in the unsupervised learning to generate $\mathcal{US}$ of SST, we will also evaluate the effect of two major parameters used in the unsupervised learning, *i.e.*, number of runs of lead clustering N_{runs} and number of top outlying training data N_{top} we choose to apply MOGA.

Effect of Pre-specified Outlier-ness thresholds

We first evaluate the effect of the pre-specified outlier-ness thresholds on the performance of SPOT. We use RD4 in this experiment and evaluate the detection rate and false positive rate under five different outlier-ness thresholds, *i.e.*, 10, 20, 30, 40 and 50. Intuitively speaking, a higher threshold will result in a lower detection rate and a lower false positive rate, and vice versa. The results are presented in Table 6.6. As

	t=10	t=20	t=30	t=40	t=50
Detection rate	99.94%	99.82%	96.77%	96.65%	96.65%
False positive rate	29.59%	20.35%	15.10%	12.43%	10.57%

Table 6.6: Performance of SPOT under varying thresholds

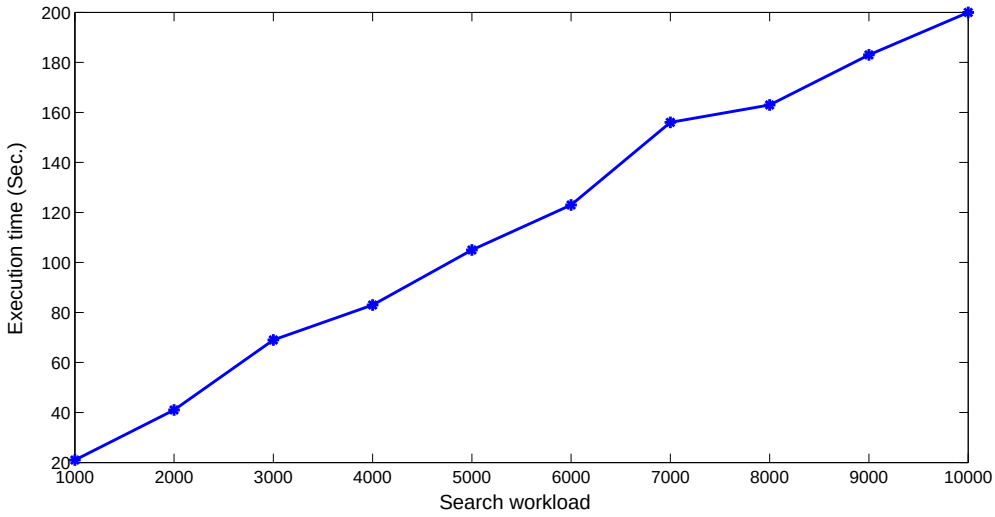

Figure 6.15: Effect of search workload on speed of MOGA

we can see from this table, SPOT is able to consistently achieve very good detection rates (over 96%) for all the five different thresholds. This indicates the insensitivity of the detection rate of SPOT with respect to the thresholds. In contrast, the false positive rate of SPOT is relatively more sensitive to the thresholds. It changes from 29.59% to 10.57% when the threshold value is reduced. However, in our applications, the detection rate is normally considered more important than the false positive rate, and a high and reliable detection rate (as achieved by SPOT) is a desired feature for the outlier detection methods in these applications.

Effect of Search Workload of MOGA W

The search workload of MOGA represents the total number of subspaces that are explored by MOGA. It directly affects the speed of MOGA and may also influence

the detection effectiveness of SPOT. If the terminating condition of MOGA is specified using the maximum number of generations performed in MOGA, then the search workload is determined by the human-specified parameters including the number of generations N_g and the number of individuals (subspaces) generated in each generation N_p. Let W denote total search workload of MOGA and it can be quantified as $W = N_g \times N_p$.

Figure 6.15 shows the execution time of the MOGA component of SPOT under varying search workloads, ranging from 1,000 to 10,000. Clearly, the execution time of MOGA increases approximately linearly with respect to its search workload. Figure 6.16 presents the values of the three optimizing criteria, RD, IRSD, IkRD, averaged over 5 different MOGA runs when the search workload is increased from 1,000 to 10,000. The result in Figure 6.16 indicates that the three criteria are steadily improved by increasing the search workload of MOGA. Nevertheless, the $\mathcal{US}$ or $\mathcal{SS}$ in SST is relatively small compared with $\mathcal{FS}$, thus MOGA is able to produce a good $\mathcal{US}$ or $\mathcal{SS}$ When a sufficiently large search workload is allowed. Once a good $\mathcal{US}$ or $\mathcal{SS}$ has been achieved, solely increasing the search workload will not lead to a significant improvement of $\mathcal{US}$ or $\mathcal{SS}$ any more. In Figure 6.16, the improvement of the three criteria when the search workload is increased from 6,000 to 10,000 is obviously much more marginal than the improvement obtained earlier when the search workload is increased from 1,000 to 6,000.

Effect of Number of Clustering Runs N_{runs}

To generate $\mathcal{US}$ in SST, multi-run leading clustering is employed to find the top outlying training data. The outlying degree of the training data are called Outlying Factor. In this experiment, we will study the change of Outlying Factor of training data when the number of clustering runs is increased. SD1, RD1, RD2 and RD3 are used in this experiment. Figure 6.17 shows the result. The value of Y axis when $X = j$ in this figure corresponds to the average difference of Outlying Factors for all training data observed in the i^{th} run with regard to the $(j-1)^{th}$ run, *i.e.*, $Y(j) = \frac{\sum_{i=1}^{N} |OF^j(p_i) - OF^{j-1}(p_i)|}{N}$. As we can see from Figure 6.17, the outlying degree

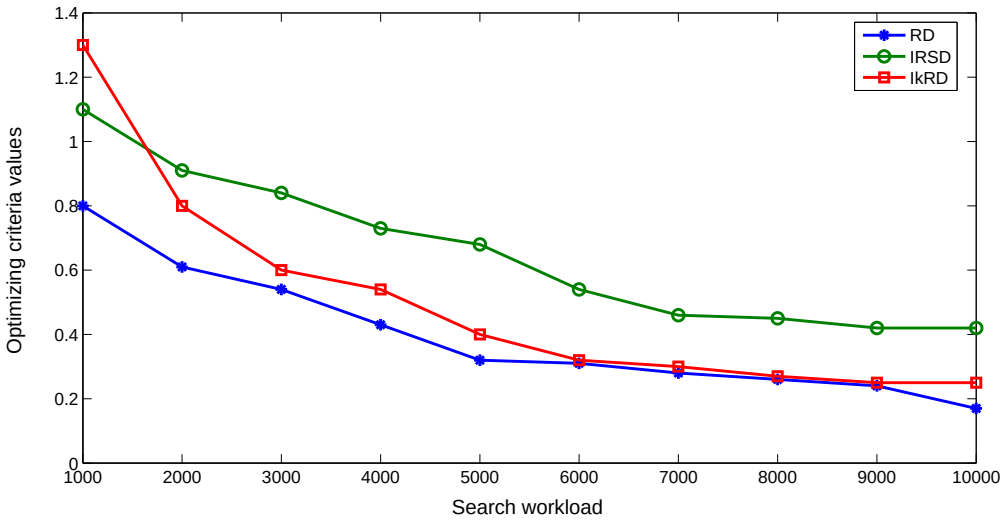

Figure 6.16: Effect of search workload on objective optimization

of training data becomes stabilized when the number of clustering runs is increased, demonstrating a good convergence phenomenon. This confirms the effectiveness of using multi-run strategy to reduce order sensitivity of the lead clustering method. Also, this experiment suggests that it is unnecessary to have a large N_{runs} value. Based on our experimental experiences, specifying N_{runs} between 5 and 10 will be generally sufficient to achieve a good convergence of Outlying Factor for the training data.

Effect of Number of Top Outlying Training Data N_{top}

The number of top outlying training data N_{top} we select may influence the effectiveness of SPOT in detecting projected outliers. Figure 6.18 presents the effect of N_{top} on SPOT using RD3 and RD4 because the ground-truth results can be established in RD3 and the labeled training data are available in RD4. We evaluate the percentage of outliers correctly flagged by SPOT using only $\mathcal{US}$ in these data sets under different N_{top} values. We can see that, as N_{top} is increased, the number of detected projected outliers increases as well but tends to converge later on. This is because $\mathcal{US}$ as well as the detecting capability of SPOT is enhanced by increasing N_{top} but approaches its

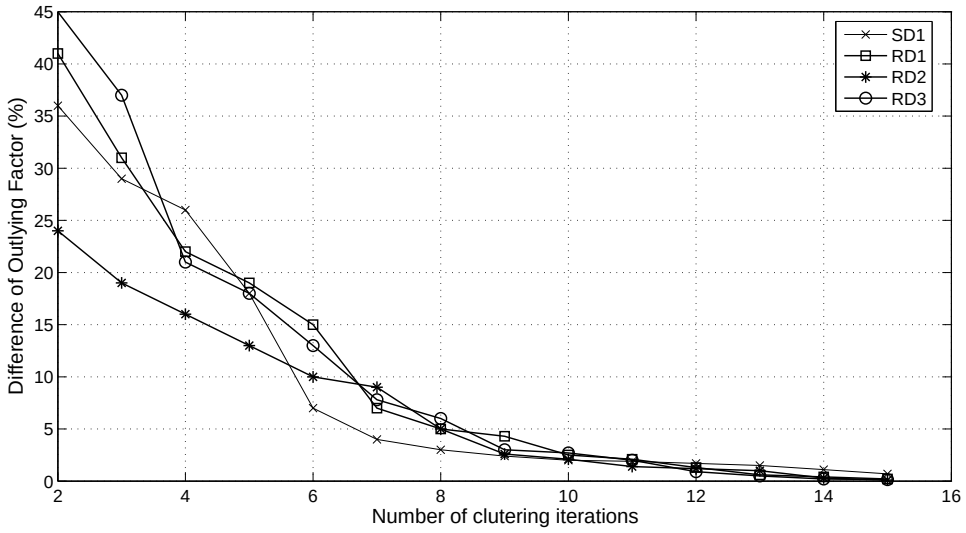

Figure 6.17: Effect of number of clustering iterations

limit when almost all the outlying training data have been included. This experiment establishes that we will still be able to achieve a good effectiveness of SPOT even when N_{top} is small relative to the number of training data. A rule of thumb for specifying N_{top} is to set it as a small percentage (for instance 1%) of the total number of training data based on the estimated frequency of outliers in the stream.

6.2.4 Effectiveness Study of SPOT

In the last part of experimental evaluation, we will conduct effectiveness study of SPOT. A number of comparative experiments will also be performed between SPOT and other competitive methods.

Competitive Methods for Comparative Study

Since there is little research conducted in projected outlier detection for high-dimensional data streams, we cannot find the techniques tackling exactly the same problem as SPOT does for comparison. However, there are some related existing approaches for detecting outliers from data streams that we can compare SPOT with. These methods can be broadly categorized as methods using *histogram*, *Kernel density function*,

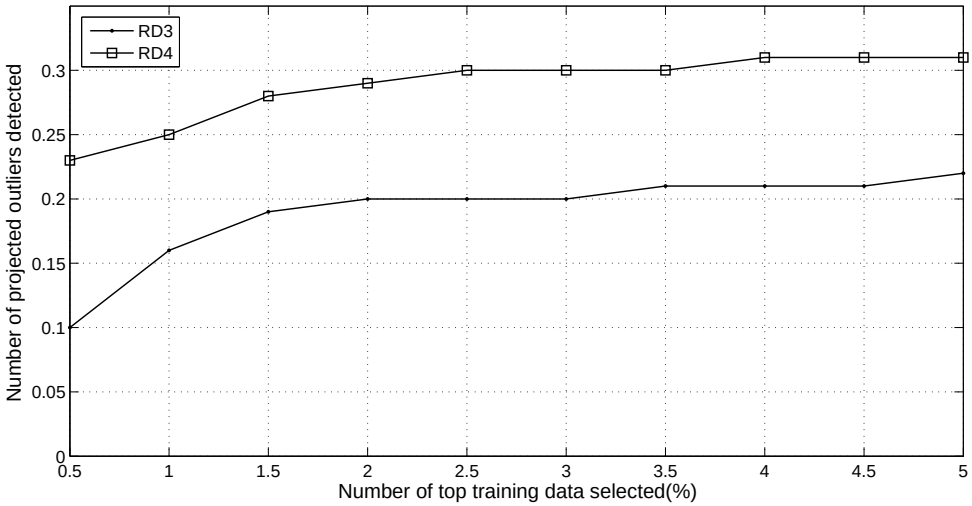

Figure 6.18: Effect of number of top outlying training data selected

distance-based function and *clustering analysis*, respectively.

Histogram and Kernel density function are amongst the most commonly used statistical techniques in outlier detection. Histograms are created for each dimension of the data stream. The density (*i.e.*, number of data points) falling into each bin of the histogram are recorded. The outlier-ness of a data point is computed feature-wise for multi-variate data as follows:

$$outlier\_ness(p) = \sum_{f \in F} w_f \times (1 - p_f)/|F|$$

where p_f is the probability that feature f takes the value of p, p_f is calculated as the ratio of the density of the bin p belongs to against the total number of points arriving thus far. w_f is the weight assigned to feature f. For simplicity, w_f is set equal for all the attributes in the data stream. All attributes are considered in calculating the outlier-ness of data points of the stream in the histogram method. $|F|$ denotes the total number of features in the data set. In this histogram-based anomaly detection method, a data point is an anomaly if its outlier-ness is above a threshold. Kernel density function models local data distribution in a single or multiple dimensions of space. One of the representative methods using Kernel function is proposed in [100] for detecting anomalies from sensor network. A point is detected as an anomaly if

the number of values that have fallen into its neighborhood (delimited by a sphere of radius r) is less than an application-specific threshold. The number of values in the neighborhood can be computed by the Kernel density function. To facilitate the comparison, the functionalities of this Kernel function based method for dealing with distributed nodes are ignored. Anomaly detection from data stream is performed only in a single-node mode for this method.

A major recent distance-based method for data stream is called *Incremental LOF* [97]. It is a variant of LOF method tailored for coping with frequent data updates (insertions and deletions) for dynamic databases.

Clustering analysis can also be used to detect outliers from those data that are located far apart from the data clusters. *HPStream* [9] is the representative method for finding subspace clusters in high-dimensional data streams. In our experiments, a minor modification is needed to enable HPStream to detect anomalies. A data will be labeled abnormal if it is far apart from the so-called limiting radius of all the clusters. Here, the limiting radius of a cluster is typically a few times of the average radius of the data points in the cluster. We test four different possible sets of dimensions associated with each cluster, *i.e.*, 1, 2, 3 and 4, respectively. The exact number of dimension for a cluster is chosen from these four configurations such that the best F-measure can be achieved. This specification ensures that all the subspaces explored in HPStream are included in the 4-dimensional space lattice.

In addition, since the Largest_cluster method [79] has been applied to the wireless network anomaly detection application, we will also compare SPOT and this method in the wireless network anomaly detection application.

Amongst the afore-mentioned competitive methods, histogram method mainly deals with each single attribute while HPStream has already been equipped with the mechanism to explore subspaces for optimizing clusters. The Kernel density function method, Incremental LOF and the Largest_Cluster method can handle multiple attributes but they lack the ability to explore subspaces. Their detection performance is thus heavily dependent on the selection of subspaces for outlier detection. In the experiments, we study two different ways for deciding the subspaces where these two

methods will be applied, namely randomly selecting multiple subspaces and using SST that is obtained by SPOT. We use SD2, RD3 and RD4 to carry out the comparative experiments. As RD3 is an unlabeled data stream, an offline detection is thus performed to establish the ground-truth results prior to the comparative study.

The wireless network anomaly detection data stream contains timestamp information for each connection. However, the existing anomaly detection methods for time-series data streams, such as [1] and [82], are not applicable in this study. This is because different users, with likely varied patterns of behaviors in using the wireless network, can access the wireless network through the same access point. Therefore, there are no evidences for supporting strong temporal dependency for data in the stream from the access point's perspective. If the time-series anomaly detection methods in [1] and [82] are used, the patterns of one user will be utilized to evaluate those of other users. This will seriously adversely affect the detection performance.

Effectiveness Measures for Comparative Study

Appropriate performance metrics are needed in evaluating the detection performance of SPOT and other competitive detection methods.

In the data sets generated using SD2 and the KDD-CUP'99 Outlier Detection application, we will use *detection rate* (or called true positive rate) and *false positive rate* for performance evaluation. These two metrics are the most commonly used onces in detection systems. Detection rate refers to the percentage of hits (correctly identified anomalies) in the whole set of anomalies existing in the data set and false positive rate represents the percentage of erroneously labeled anomalies in the whole set of normal data. Their definitions are presented as follows.

$$Detection\ rate = \frac{|\{anomalies\ correctly\ detected\ by\ the\ detection\ method\}|}{|\{true\ anomalies\}|}$$

$$False\ positive\ rate = \frac{|\{normal\ instances\ erroneously\ labeled\ as\ attacks\ by\ the\ method\}|}{|\{true\ normal\ instances\}|}$$

Based on detection rate and false positive rate, Receiver Operating Characteristic (ROC) analysis is usually used. ROC is a commonly used technique for performance evaluation of detection methods by plotting its detection rate and false positive rate

in the same ROC space. In a ROC curve, the detection rate is plotted in function of the false positive rate for different cut-offs. The closer the ROC curve of a method is from the left-upper corner in ROC space, the better the detection performance of this method will be.

In the wireless network anomaly detection application, we use *precision* and *recall*, rather than detection rate and false positive rate. Precision and recall are two metrics originally used to evaluate the performance of information retrieval systems. They are also widely used for performance evaluation of detection systems. Precision measures accuracy of the detection results while recall measures completeness of the detection results. The reason why we use precision-recall analysis, instead of ROC analysis, is because we need to know all the true normal data in order to compute false positive rate in ROC analysis. This requires the validation of almost all the data in the data stream for generating the ground-truth result given the majority of the data in the data stream are normal. This will be extremely expensive as the validation for each data is rather costly. In contrast, Precision-Recall analysis only needs the true anomalies in the data stream. We can validate the anomalies flagged out by different detection methods in order to find the anomalies in the data stream. We understand that this is only an approximate approach for finding the true anomalies in the data streams, but in practice it works well because the true anomalies are quite likely to be detected by at least one of the methods under study. As the number of anomalies is much smaller than the number of normal data in the stream, Precision-Recall analysis becomes more efficient than ROC analysis. Additionally, the underlying relationship between ROC space and Precision-Recall space has been unveiled recently in [37], that is, a curve dominates in ROC space if and only if it dominates in Precision-Recall space. Therefore, if ROC analysis is computationally infeasible, then we can perform evaluation based on the precision and recall criteria as an equivalent alternative strategy.

In an outlier detection application, precision is defined as the fraction of anomalies that the detection method correctly identifies, *i.e.*,

$$Precision = \frac{|\{anomalies\ correctly\ detected\ by\ the\ detection\ method\}|}{|\{anomalies\ detected\ by\ the\ detection\ method\}|}$$

	SPOT	Histogram	Kernel function	Incremental LOF	HPStream
Detection rate	100%	0.04%	100%	100%	0.29%
False positive rate	0%	0%	2.5%	0%	0%

Table 6.7: Comparison of different methods using SD2

Recall is defined as the fraction of true anomalies that are successfully identified by the detection method, *i.e.*,

$$Recall = \frac{|\{anomalies\ correctly\ detected\ by\ the\ detection\ method\}|}{|\{true\ anomalies\}|}$$

F-measure is the metric combining precision and recall into a single measure for performance evaluation. It is defined as

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Experimental Results Using Synthetic Data Set SD2

We first use the synthetic data sets generated by SD2 to carry out comparative experiment between SPOT and other four competitive methods. Unlike SD1, SD2 is able to facilitate the performance analysis as the true projected outliers in SD2 are known in advance. The performance of different methods is measured by detection rate and false positive rate.

As illustrated in Table 6.7, we can see that SPOT, Kernel function method and Incremental LOF are able to detect all the projected outliers in SD2 (featuring a 100% detection rate), while histogram method and HPStream have a very poor detection rate. Histogram employs feature-wise analysis for detecting outliers. All the outliers in SD2 appear perfectly normal in each dimension because they have been masked by the normal data, thus the histogram method is not able to identify them. The ability of HPStream in detecting outliers is largely limited by the small number of subspaces it explores. All the methods perform well in terms of false positive rate due to the relatively simple structure of the data sets generated by SD2.

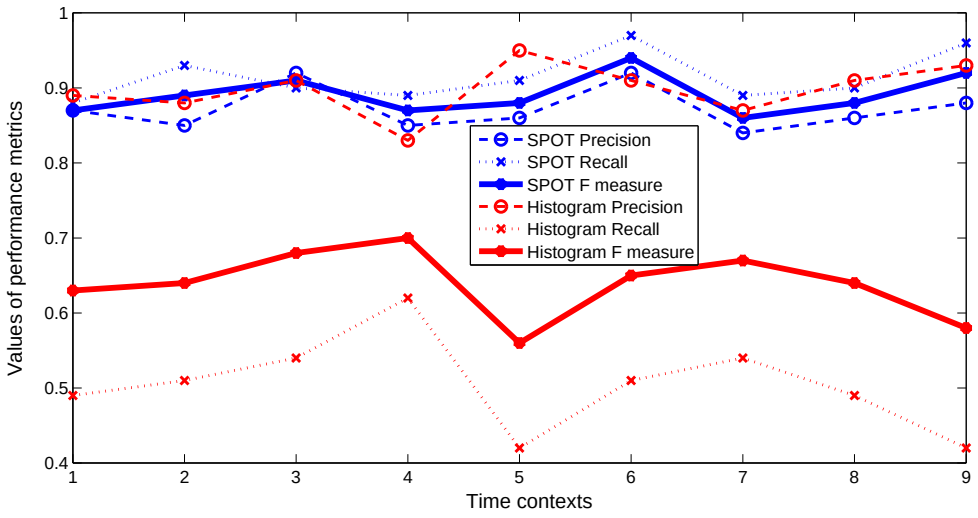

Figure 6.19: Precision, recall and F-measure of SPOT and the histogram based method

Experimental Results Using Wireless Network Anomaly Detection Data Stream

In this section, we conduct experimental evaluation on SPOT for anomaly detection from the wireless network data stream. Extensive comparative study are performed to investigate the detection performance between SPOT and the competitive anomaly detection methods.

- **Comparative Study Between SPOT and Anomaly Detection Method Using Histogram Technique**

In this first experiment, we will compare SPOT and anomaly detection method using histogram technique.

Figure 6.19 presents the detection performance of SPOT and the histogram method. The outlier-ness threshold for the anomaly detection method using histogram technique is set as 0.8. From the figure, we can see that the histogram-based anomaly detection method achieves a slightly better precision performance but a significantly low recall performance when compared with SPOT. The F-measure of the histogram-based anomaly detection technique is also inferior to that of SPOT due to its poor

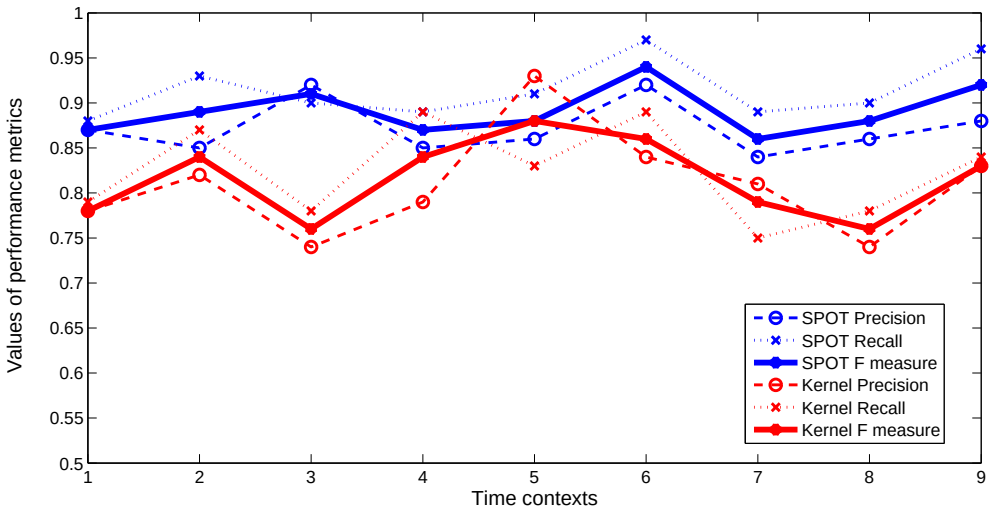

Figure 6.20: Precision, recall and F-measure of SPOT and the Kernel-function based method

recall performance. This is because that the histogram-based anomaly detection method is only able to identify anomalies that exhibit outlier-ness in all or most attributes. The method cannot successfully detect those anomalies that are outlying only in a small number of attributes but their overall outlier-ness is not sufficiently high in order for them to be labeled as anomalies.

• Comparative Study Between SPOT, Kernel Function based Anomaly Detection Method and Incremental LOF

In this experiment, we compare SPOT, the Kernel function based anomaly detection method and Incremental LOF. The reason why we compare these two methods together with SPOT is that neither the Kernel function based anomaly detection method nor Incremental LOF has the ability to explore different subspaces for detecting anomalies. They both need human users to specify subspaces for performing anomaly detection.

The comparative study between SPOT, the Kernel function based method and Incremental LOF are carried out as follows:

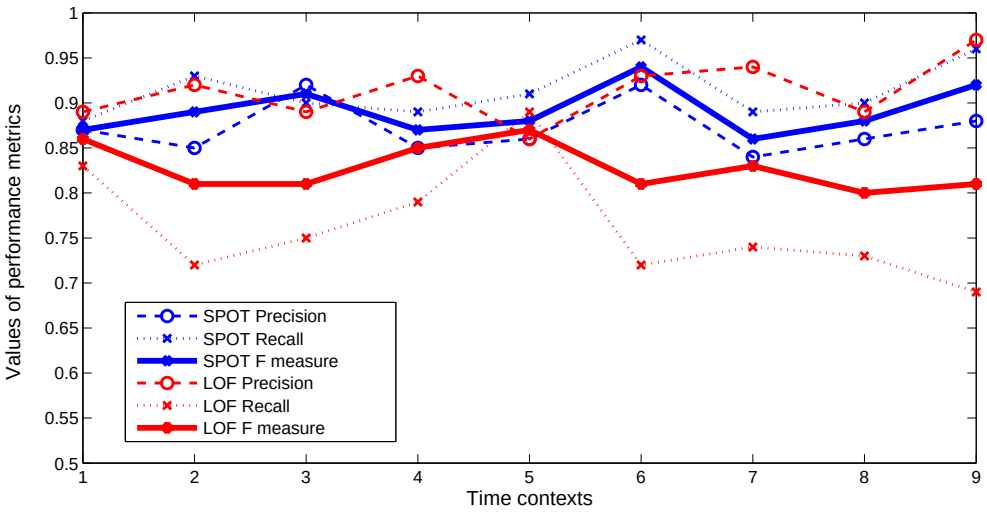

Figure 6.21: Precision, recall and F-measure of SPOT and the Incremental LOF

1. We first study the detection performance of SPOT and the other two competitive methods under the same searching subspaces. Specifically, we compare their performance in subspaces of SST. Figure 6.20 shows the results of SPOT and the Kernel function method. We can see that the performance of SPOT is noticeably superior to the Kernel-function based method; SPOT achieves better precision in 8 out of 9 temporal contexts and better recall and F-measure in all temporal contexts. Kernel density estimation is essentially a local density method. As it can quantitize the number of data points within a specific neighborhood through Kernel function, thus Kernel function method is similar in spirit to the $D(k, \lambda)$-Outlier method, with the difference that the Kernel function can be incrementally updated while $D(k, \lambda)$ metric cannot. SPOT does not only use local density metrics but also global distance metrics in the anomaly detection, which contributes to a better detection performance. Figure 6.21 presents the results of SPOT and Incremental LOF. Detection accuracy of Incremental LOF is better than SPOT in 6 out of 9 temporal contexts. Nevertheless, SPOT features better recall and F-measure than Incremental LOF in all the temporal contexts. Incremental LOF can achieve a better detection precision than SPOT because it employs LOF, a much more complicate metric, to

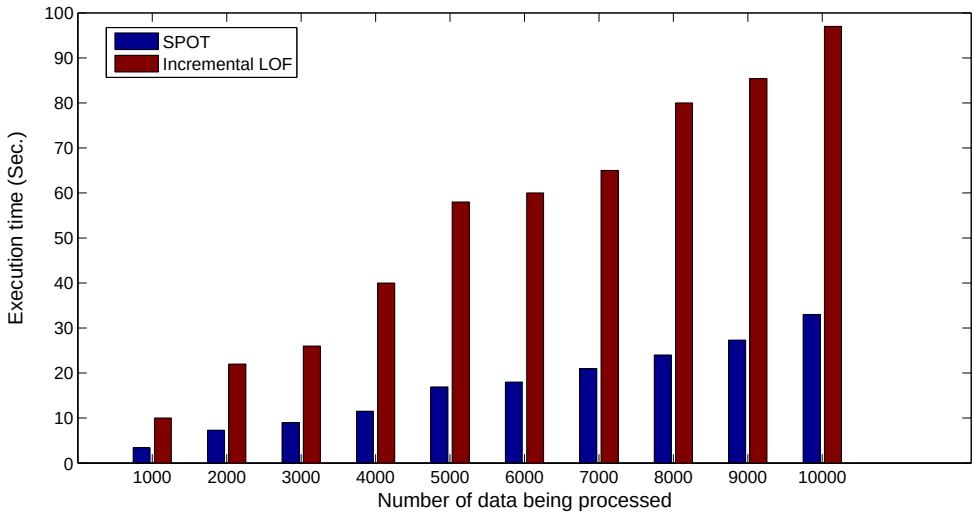

Figure 6.22: Efficiency comparison of SPOT and Incremental LOF

detect anomalies. As a result, Incremental LOF is subject to a far higher computational overhead than SPOT. Figure 6.22 presents the execution time of SPOT and Incremental LOF under varying number of data to be processed. SPOT is obviously much more efficient than Incremental LOF in processing data in the data stream. Better efficiency is especially important in data stream applications. Moreover, unlike maintaining only a compact data synopsis in SPOT, Incremental LOF needs to keep all the data in memory, leading to an extremely high space overhead.

2. The second experiment we conduct aims to evaluate the detection performance of the three methods under different searching subspaces. SPOT detects anomalies in SST while the other two competitive methods detect anomalies in a specific number of randomly chosen subspaces. In other words, the subspaces explored by the Kernel function based method and Incremental LOF are chosen entirely in a random manner, while those of SPOT are driven by its own fitness function. For a fair comparison, the total search workload is set identical for the three methods. We evaluate the number of anomalies these three methods can correctly detect under different search workloads (*i.e.*, the number of subspaces

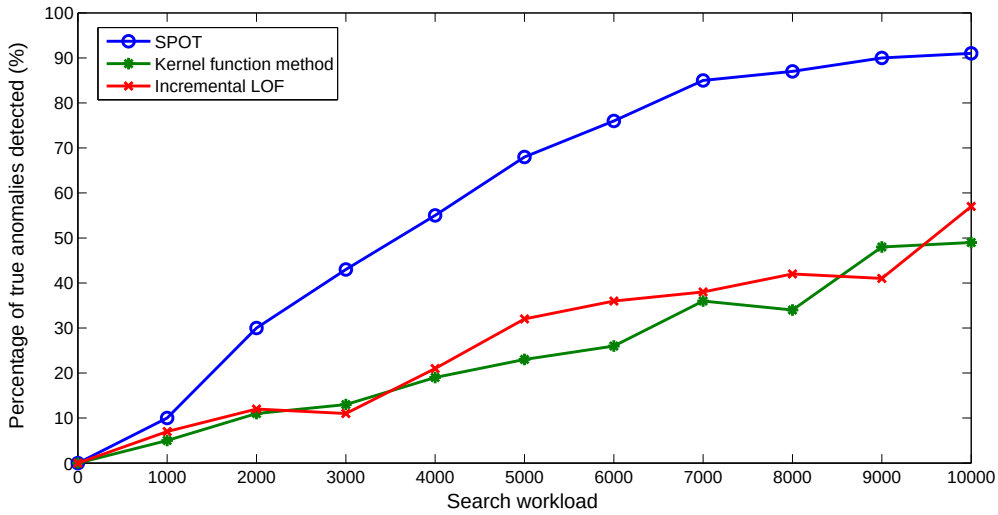

Figure 6.23: Percentage of true anomalies detected by SPOT, the Kernel function-based detection method and Incremental LOF under varying search workloads

to be evaluated). The search workloads are specified from 1,000 to 10,000 with 1,000 being increments. The emphasis of this study is to see how comprehensive the three methods can detect anomalies. Figure 6.23 shows the number of anomalies (in percentage) correctly detected by the three methods under different subspaces. The results are averaged on all the 9 temporal contexts. We can see that from the figure that, under the same searching workload, the number of detected anomalies of SPOT is significantly (approximately 35%-43%) higher than the other two methods when the search workload reaches 10,000. This indicates a noticeable better detection capability of SPOT given a fixed search workload. This advantage of SPOT is due to the fact that SPOT is equipped with the ability of automatic subspace search by means of MOGA while the other two methods do not. A lack of the ability for guided subspace searching will seriously limit the Kernel-function based method and Incremental LOF in detecting anomalies in high-dimensional data space;

3. It is worthwhile pointing out that the Kernel-function based method and Incremental LOF, including most of other anomaly detection methods, typically operate in *a single human pre-specified data space* for detecting anomalies. If

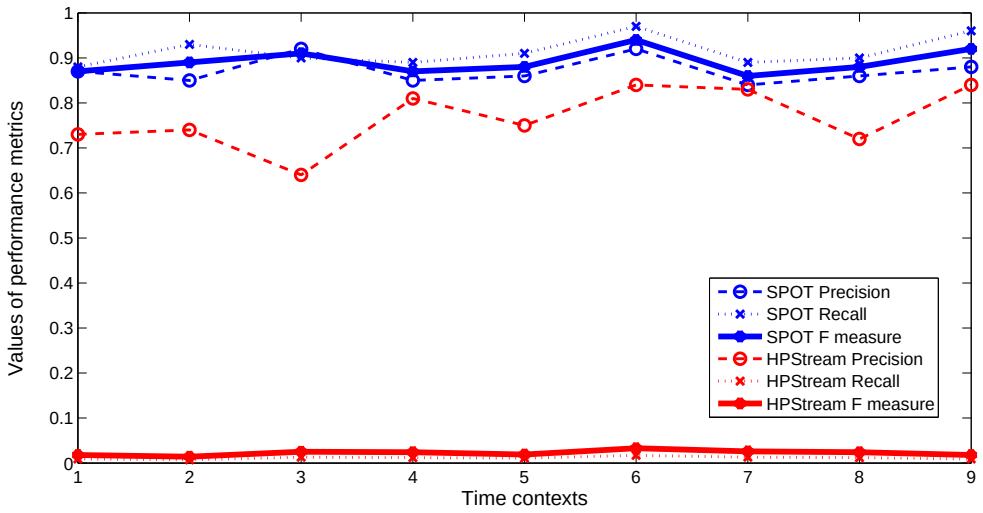

Figure 6.24: Precision, recall and F-measure of SPOT and HPStream

this practice is used, then the number of anomalies that can be detected by the Kernel-function based method and Incremental LOF in any single subspace is at most 0.18% and 0.46%, respectively, of the total number of anomalies that can be detected by SPOT, suggsting that they will fail to detect a significant portion of the ture anomalies.

• Comparative Study Between SPOT and Projected Clustering Method for High-dimensional Data Streams

Given the possibility of utilizing clustering method in high-dimensional data streams, such as HPStream, in anomaly detection, we would like to carry out a comparative study between SPOT and HPStream.

The precision, recall and F-measure results of SPOT and HPStream are presented in Figure 6.24. The results suggest the following two major findings:

1. The precision of HPStream is worse than that of SPOT. As we know, precision performance is largely dependent on the outlier-ness metric the detection method uses. In HPStream, a data point is detected as abnormal in a subspace if

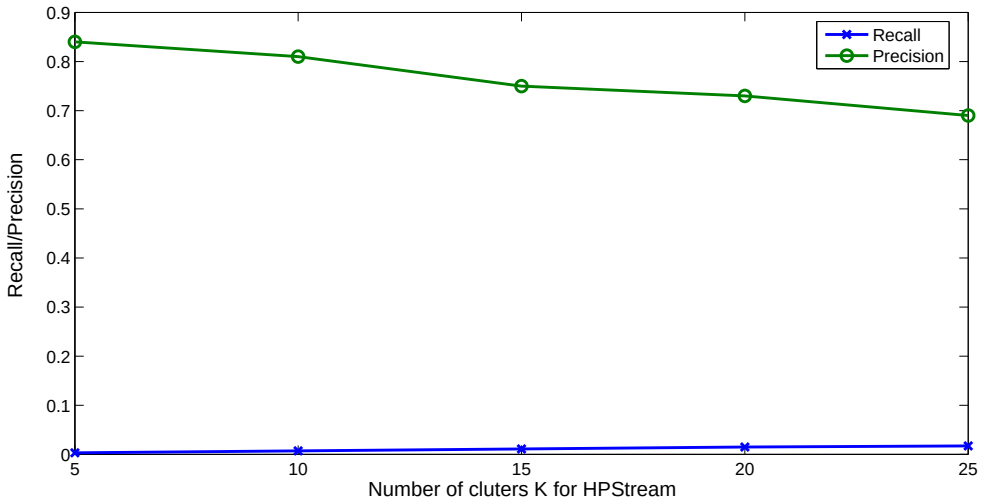

Figure 6.25: Precision and recall of HPStream under a varying number of clusters

it is far from the limiting radius of the clusters that can be optimized in this subspace. This judgment, to a large extent, is similar to the Largest_Cluster method given that the possibility that two or more clusters can be optimized in the same subspace is quite low. Hence, the precision of HPStream is inferior to SPOT. SPOT utilizes more complicated outlier-ness metrics than HPStream. These metrics help SPOT achieve a higher level of accuracy in identifying anomalies;

2. HPStream features a much poorer recall performance compared to SPOT. The major reason for the low recall performance of HPStream is because of the extremely small number of subspaces where anomalies can be detected. As only one set of dimensions is associated with each of the k clusters, there will be only k subspaces where the newly arrived data points are evaluated to decide whether they are abnormal or not. It is impossible for HPStream to evaluate the newly arrived data in the subspaces other than those associated with the clusters that HPStream has produced. This could result in a large number of anomalies being missed out in the detection result.

Because the number of clusters k to be obtained is an important parameter in HPStream, we also study the effect of k in this experiment. We test five different

reasonable values of k for HPStream, *i.e.*, 5, 10, 15, 20 and 25, respectively. The corresponding precision and recall are plotted in Figure 6.25. We can see from this figure that the recall of HPStream is improved as k increases. This improvement of recall is due to the larger number of subspaces (resulting from the larger number of clusters) that can be used in anomaly detection. Because of the recall-precision dilemma, the precision of HPStream tends to become lower under an improving recall. However, the improved recall is still far from satisfactory because 25 subspaces, in any sense, is too small even compared with the possible subspaces in a low-dimensional space lattice. One may argue that the recall of detection can be greatly improved simply by increasing the number of clusters k. However, an unreasonably large k will cause the clusters obtained to be inconsistent with human perception and render many data being erroneously labeled as anomalies.

Based on this experiment, we learn that even though HPStream is able to deliver high-quality clusters embedded in subspaces, it is not trivial for it to achieve a satisfactory anomaly detection performance with only a minor modification. This is due to the discrepancy of the objective pursued by SPOT and HPStream: SPOT tries to deliver a good detection of anomalies, while HPStream seeks to find compact clusters in some subspaces. This experiment suggests that it is quite difficult, if it is not impossible, for the existing clustering methods in high-dimensional data streams to provide a sufficient support to the problem of anomaly detection in high-dimensional data streams.

• Comparative Study Between SPOT and *Largest_Cluster* Detection Method

The anomaly detection method that employs the *Largest_Cluster* metric for detecting anomalies [79] uses this wireless data stream as a case study. Therefore, we would like to carry out a comparative study between SPOT and the Largest_Cluster method.

As we have discussed earlier, the Largest_Cluster detection method is not able to directly deal with data streams, it is thus impossible for us to produce a set of anomalies from the data stream by this method. To overcome this problem and enable us to compare SPOT and the Largest_Cluster detection method in a reasonable

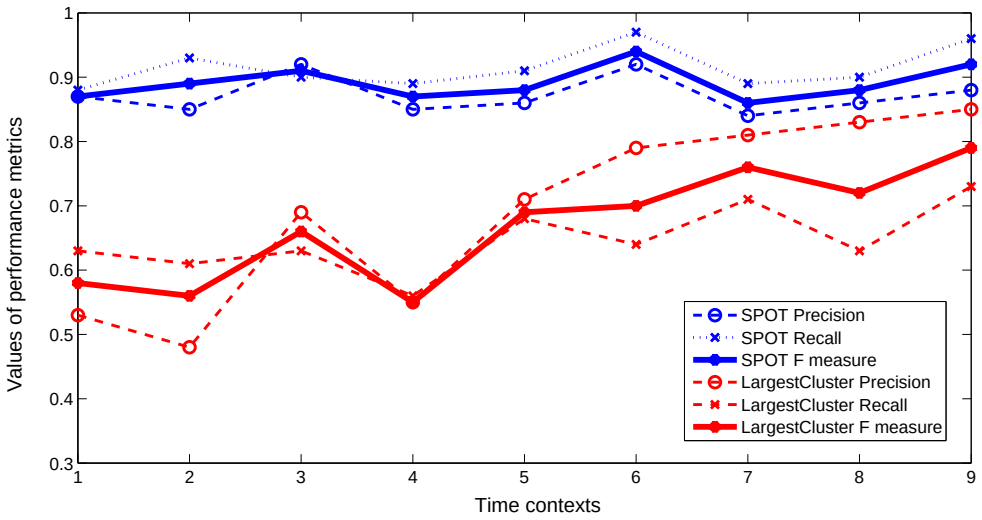

Figure 6.26: Precision, recall and F-measure of SPOT and the Largest-Cluster detection method

way, a minor modification of the Largest_Cluster detection method is conducted. We employ lead clustering method, instead of k-means, to cluster data efficiently. In lead clustering, each newly arrived data will be assigned to the largest cluster generated thus far using the data arriving earlier. The centroid of the cluster to which the newly arrived data is assigned will be updated. This modification enables the detection method to process the data in an incremental fashion while maintaining the spirit of the Largest_Cluster metric in anomaly detection.

We first evaluate the average precision, recall and F-measure of SPOT and the Largest_Cluster detection method under a fixed SST. Due to the high sensitivity of the Largest_Cluster detection method to the distance cutoff threshold α and number of clusters k, five different pairs for k and α are used in this experiments, *i.e.* $\{k = 5, \alpha = 1.0\}$, $\{k = 10, \alpha = 2.0\}$, $\{k = 15, \alpha = 3.0\}$, $\{k = 20, \alpha = 4.0\}$ and $\{k = 25, \alpha = 5.0\}$. A general rule for paring k and α is that a smaller k value is paired with a smaller α value and vice versa. The precision-recall results are shown in Figure 6.26. From this figure, we can see that SPOT is able to reliably achieve a good detection performance in different temporal contexts. The F-measure achieved by SPOT is in the range of 0.79-0.89 for different temporal contexts. This indicates a

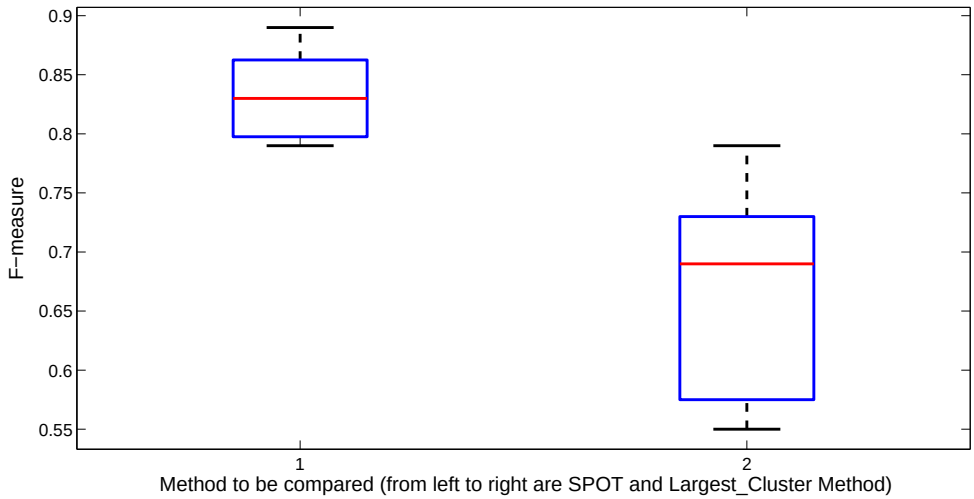

Figure 6.27: Boxplot of F-measure of SPOT and the Largest-Cluster detection method

good insensitivity of SPOT towards different temporal contexts. In contrast, the performance of the Largest_Cluster detection method varies noticeably under different temporal contexts. Its F-measure ranges from 0.55 to 0.79. For a better comparison, please see Figure 6.27 for a boxplot comparison of the F-measures of the two methods. A closer study reveals that the detection performance of the Largest_Cluster detection method tends to deteriorate when it is performed in relatively high-level temporal contexts. Specifically, the F-measure in temporal context of *All* is generally worse than that of single temporal contexts and the detection accuracy of *All* and single-dimensional temporal contexts are inferior to that of two-dimensional temporal contexts. This is because that, as data are generalized (when we roll up the temporal contexts along the lattice), they tend to display a multi-mode distribution in different subspaces, meaning that multiple natural clusters may be formed and some normal data may not be located in the limited neighborhood of the largest cluster. This will greatly pose an adverse effect on the performance of the Largest_Cluster detection method in correctly identifying anomalies.

We also study the F-measure values for the two methods under different number of search subspaces. Precisely, we change the size of SST from 1,000 to 10,000, and

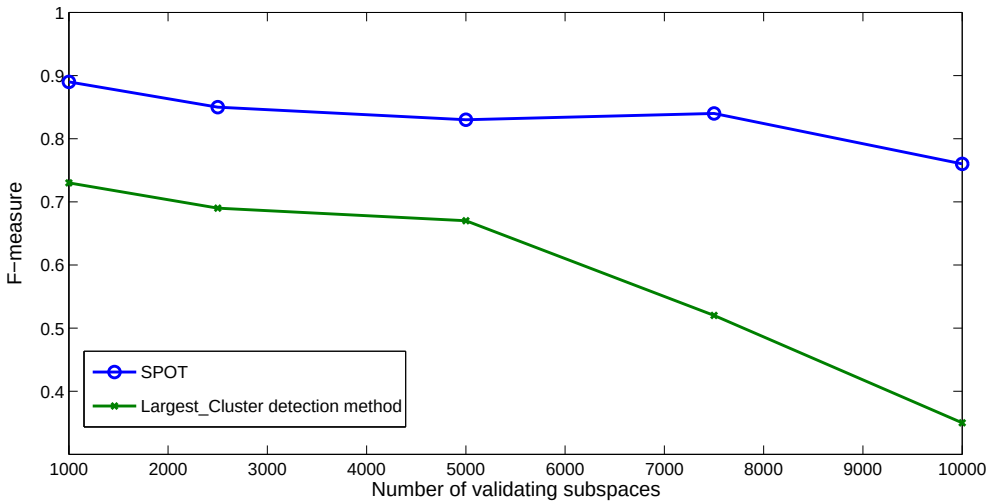

Figure 6.28: F-measure of SPOT and the Largest-Cluster detection method under varying number of validation subspaces

investigate the F-measure values for the two methods under varying SST sizes. The results are presented in Figure 6.28. In this figure, we show the F-measure values of SPOT and Largest_Cluster detection method averaged over the 9 different temporal contexts. We found from this figure that SPOT outperforms Largest_Cluster detection method in F-measure performance. In addition, the SST size does not seem to be an affecting factor to the detection performance of SPOT, while the Largest_Cluster detection method is much more sensitive to the SST size. We witness a sharp decrease in F-measure value for the Largest_Cluster detection method when SST is increased. This is due to the fact that when anomalies are to be detected from an increasingly large number of subspaces, there is a higher chance that the values of parameters k and α used in the Largest_Cluster method become inappropriate for anomaly detection in many subspaces. Specifying ad-hoc parameter values for k and α in different subspaces will be an overwhelming task that is infeasible in practice. In comparison, SPOT employs ratio-type parameters that are nearly independent of the data scale in different subspaces, leading to a more reliable F-measure performance.

	All	Daytime	Night	Weekday	Weekend
All	X	X	X	X	X
Daytime	15%	X	X	X	X
Night	13%	X	X	X	X
Weekday	19%	X	X	X	X
Weekend	24%	X	X	X	X
Weekday-Daytime	28%	26%	X	22%	X
Weekday-Night	38%	X	22%	38%	X
Weekend-Daytime	42%	17%	X	X	23%
Weekend-night	31%	X	16%	X	20%

Table 6.8: Anomaly detection analysis for different temporal contexts

• **Validating Data Partitioning of the Stream**

Recall that when we apply SPOT to the wireless network anomaly detection application, data in the stream are partitioned according to different temporal contexts. The motivation for doing this is that there may be anomalies that can be detected in more specific temporal contexts such as *Weekday-Daytime* but cannot be detected in its more general temporal contexts such as *All*, *Weekday* or *Daytime*. This is due to the different data distribution/characteristics in temporal contexts of varying granularities. This experimental analysis tries to establish the benefit of partitioning the data stream into several sub-streams in an attempt to detect more anomalies.

The results are shown in Table 6.8. Each entry (i, j) in this figure represents the percentage of anomalies that can be detected in temporal context i but cannot be detected in temporal context j subject to the constraint that i is a more specific temporal context of j in the temporal context lattice.

We note that no results are presented in the figure for the cases that temporal context i is identical to j and i is a more general temporal context of j. The corresponding entries are marked using symbol 'X' in the table. The reasons are given as follows:

1. If temporal context i is identical to j, then it is trivial to know that the result will be 100%;

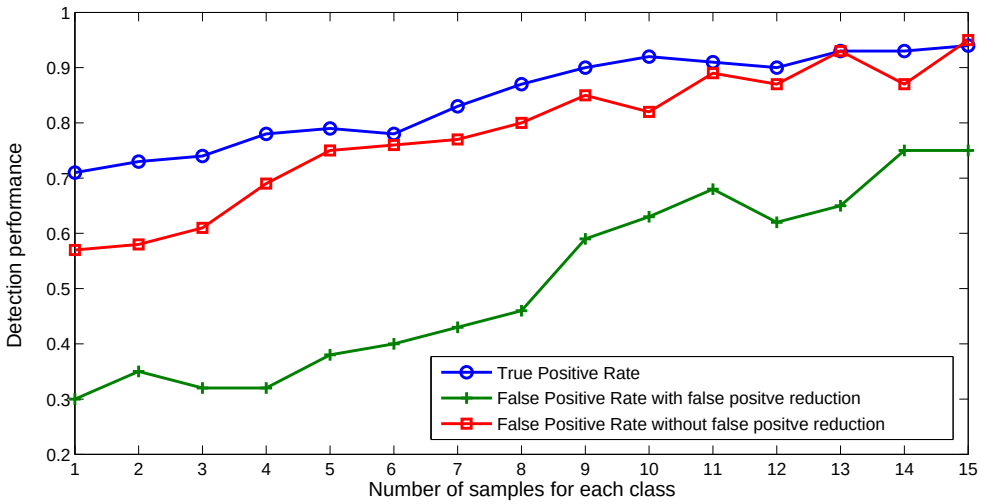

Figure 6.29: Effect of number of training data sets for each attack class

2. if i is a more general temporal context of j, then the analysis becomes meaningless. As we know, the data stream corresponds to j is only a subset of that for i, therefore the data that can be detected as anomalies in temporal context i may not even exist in the data for temporal context j at all.

From Table 6.8, we can see that on average 17.7% of anomalies in median-level temproal contexts cannot be detected if only the data of top-level temproal context is used. Further, on avergae 26.9% of the anomalies existing in the bottom-level temporal contexts will be missing if only data of median-level or top-level temporal contexts are employed. This shows the benefit of partitioning the data stream based on different temporal contexts for detecting more anomalies.

Experimental Results Using Network Anomaly Detection Data Stream

In this subsection, we report the results of experimental evaluation on SPOT in KDD-CUP'99 Outlier Detection application.

Datasets	5.0	4.0	3.0	2.0	1.0
SD1	62%	69%	72%	76%	79%
RD1	42%	55%	58%	63%	77%
RD2	44%	49%	53%	72%	84%
RD4	51%	54%	56%	59%	81%

Table 6.9: Percentage of the anomalies that have redundant outlying subspaces

• **Effect of Number of Training Data Sets**

When SPOT is applied in KDD-CUP'99 outlier detection application, multiple training data sets are generated for each attack classes for training purpose. This is to sample an enough amount of attack instances and at the same time satisfy the distribution requirement regarding normal and attack instances in each training data set. In this experiment, we investigate the effect of the number of training data sets generated for each class on the detection performance of SPOT. Recall that, due to the size limitation, each training data set is only able to contain a small portion of the labeled outlier exemplars from the original training data set. Therefore, it is expected that, as the number of training data set for each class is increased, the detection accuracy will be enhanced accordingly and finally a kind of convergence phenomenon is expected to be observed. In this experiment, we evaluate the true positive rate and false positive rate of SPOT under varying number of training data sets. The result is presented in Figure 6.29. Besides the curve of true positive rate, two additional curves corresponding respectively to the cases of using and not using the false positive reduction are also presented for the false positive rate in the figure. We can see that, as the number of training data set increases, the true positive rate is indeed improve. However, a larger number of training data sets tend to result in a higher false positive rate. Fortunately, we observe an noticeably lower false positive rate for SPOT thanks to the false positive categorization we introduced in SPOT to automatically screen out false positives through the anomaly categorization process.

Datasets	5.0	4.0	3.0	2.0	1.0
SD1	54%	62%	62%	64%	69%
RD1	49%	52%	59%	68%	70%
RD2	57%	61%	69%	70%	73%
RD4	55%	56%	57%	60%	66%

Table 6.10: Redundancy Ratio of different data sets

• **Redundant Outlying Subspace Removal**

We also investigate the existence of redundant outlying subspaces of anomalies. We first study the percentage of anomalies that have redundant outlying subspaces in KDD-CUP'99 outlier detection data stream and other data sets. We can see from Table 6.9 that the majority of anomalies have redundant subspaces (ranging from 42% to 84%). We also study the Redundancy Ratio for these data sets. Here, the Redundancy Ratio (RR) of a data set D is defined as the ratio of the number of outlying subspaces (OS) of anomalies against the size of their outlying subspace front (OSF), *i.e.*

$$RR(D) = \frac{\sum |OS(o)|}{\sum |OSF(o)|}, \text{ for } o \text{ being an anomaly in } D$$

As shown in Table 6.10, from the whole data set's perspective, the Redundancy Ratio of its outlying subspace set is between 49% and 73%, indicating a fairly high Redudency Ratio for different data sets. As a result, using Outlying Subspace Front would help reduce the number of outlying subspaces by from 49% to 73%. Another important observation is that the values of these two measures are increased when the outlier-ness threshold goes down. This is because that, as the outlier-ness threshold become smaller, more subspaces will become outlying subspaces and they are likely to be dominated by some of their lower dimensional counterparts.

• **Comparison of Manual and Automatic Methods for Identifying False Positives**

In this experiment, we compare the signature subspaces and the classification accuracy obtained by the manual and automatic methods for screening false positives.

	0.5	0.6	0.7	0.8	0.9
Difference of signature subspaces generated	23.5%	7.6%	13.7%	17.8%	28.5%
Accuracy in classifying false positives	87.2%	92.3%	88.5%	83.4%	72.1%

Table 6.11: Comparison of the manual and automatic methods for identifying false positives

The manual method draws on human expertise to identify false positives, while the automatic method automatically labels those anomalies that cannot be categorized into any known attack classes as false positives. That is, a data is labeled as a false positive if its class membership probability with respect to any attack class is lower than τ. We use a portion of the labeled training data in this experiment. The labeled training data can be used to simulate the detection results of the manual method. The results in this experiment are evaluated under varying membership probability threshold values (*i.e.*, τ). Table 6.11 shows the difference (presented in percentage) of the signature subspaces generated by these two methods for the false-positive class and the classification accuracy of false-positives using the automatic method under varying class membership probability threshold τ. This comparison indicates that the final signature subspaces of the false-positive class generated by the automatic method is similar to those generated by the manual detection method, differing marginally (less than 10%) when τ is set 0.6. In addition, using the results of the manual detection method as a reference, the classification of false positives using the automatic method achieves an accuracy around 90% when τ is set between 0.5 and 0.7. These results are attributed to the fact that most of the anomalies that are dissimilar to any existing attack classes are false positives. Consequently, it does not seriously matter if all these anomalies are directly labeled as false positives without undergoing a human screening process. This suggests that relying solely on the automatic detection for false positives is an acceptable alternative if human involvement is impossible or limited in the course of false-positive examination.

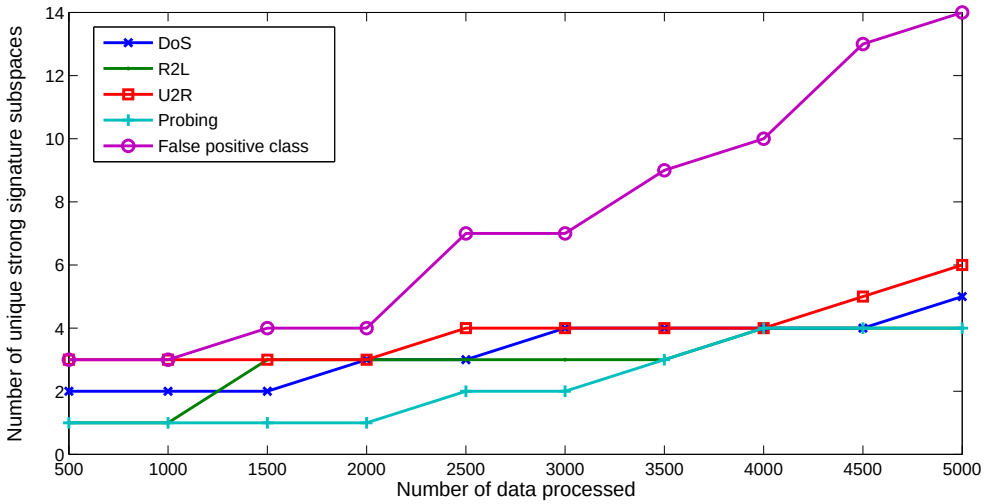

Figure 6.30: Number of strong signature subspaces for each attack class under varying number of data being processed

- **Signature Subspace Analysis**

We are also interested in studying the diversity of signature subspaces of the false-positive class, as compared with those of the attack classes. We record the number of strong unique signature subspaces for different classes (including the false-positive class) as the number of data we evaluated increases. In this experiment, the strong signature subspaces we select are those signature subspaces whose tf-idf weight is 5 times higher than the average weight level. This definition of strong signature subspaces is of course subjective. We plot the results in Figure 6.30. Interestingly, we find that the number of unique strong signature subspaces for the false-positive class is significantly higher than any other attack classes by a factor of three or four. This means that the strong signature subspaces for the false-positive class is far more diverse that those of the attack classes. This finding offers an important insight to us when we are creating the set of signature subspaces of the false-positive class. We need to collect a relatively large pool of signature subspaces in the detection process to achieve an accurate detection of false positives.

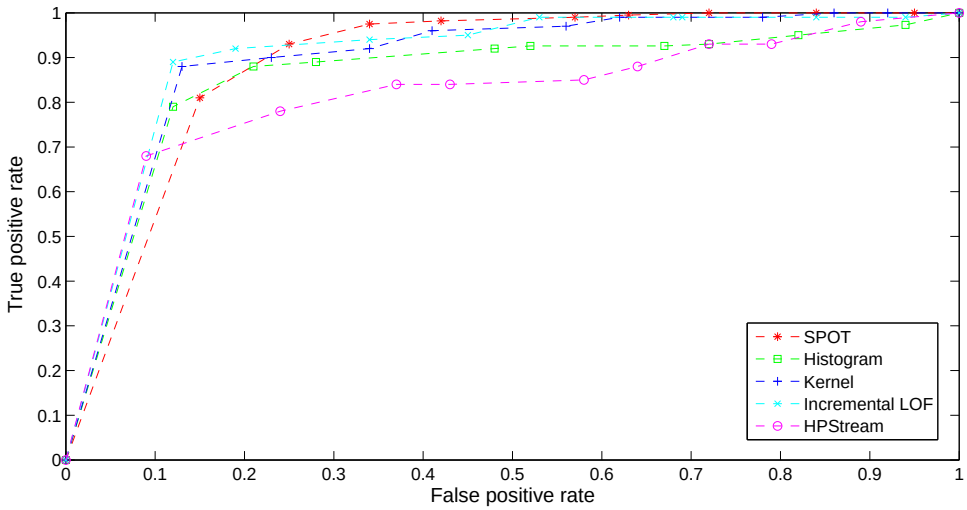

Figure 6.31: ROC curves of different methods

• **Comparative Study with Existing Methods**

Comparative study is also performed in KDD-CUP'99 outlier detection application to investigate the detection rate and false positive rate of SPOT and other existing anomaly detection methods, including histogram method, Kernel function method, Incremental LOF and HPStream. Receiver Operating Characteristic (ROC) analysis is used in this comparative study. To conduct ROC analysis, we need to know apriori the true positives (anomalies) and true negatives (normal data). This is possible in KDD-CUP'99 outlier detection application as labeled test data are available. As we know, any detection method can easily achieve a 100% detection rate by simplying labeling all the connections as anomalies. However, this strategy will result in an extermaly high false positive rate. Likewise, one can obtain a 0% false positive rate by claming all the connections as normal, but this will lead to 0% true positive rate. Therefore, we need to consider these two rates simultaneously. In Figure 6.31, we plot the ROC curves for SPOT and other four competitive methods. We can see from this figure that the ROC curves of SPOT, Incremental LOF and Kernel function method progress much closer to the upper-left coner of the plot than the curves of histogram method and HPStream, indicating that SPOT, Incremental LOF and Kernel function

method generally achieve a better detection performance. A closer examination of the figure suggests that Incremental LOF and Kernel function method perform better than SPOT when the false positive rate is relatively low (*i.e.*, in the early stage of the ROC curves). However, SPOT starts to outperform Incremental LOF and Kernel function method as the false positive rate further increases. The false positive categorization that SPOT is equipped with enables it to identify false positives in an automated fashion while other competitive methods cannot. This helps SPOT to significantly reduce false positives and achieves a lower false positive rate under the same detection rate (or achieves a higher detection rate under the same false positive rate).

• **Comparative Study with the Winning Entry of KDD CUP'99**

At the end of this subsection, we would like to compare the detection performance of SPOT with that of the winning entry of KDD CUP'99. The winning entry, due to Dr. Bernhard Pfahringer of the Austrian Research Institute for Artificial Intelligence, uses bagged boosting of C5.0 (See5) decision trees. We compare these two methods using detection rate and false positive rate. These two rates of the winning entry can be directly obtained from http://www-cse.ucsd.edu/users/elkan/clresults.html. Since there is only one set of detection rate and false positive rate published for the winning entry from this link, we tune the pre-specified thresholds of SPOT to generate a set of detection rate and false positive rate for SPOT that are as close to those of the winning entry as possible for a better comparison. Table 6.12 presents the results. The detection rate and false positive rate of the winning entry are 91% and 0.55%, respectively. The closest set of the detection rate and the false positive rate that we can obtain for SPOT are 92.3% and 1.5%, respectively, which are comparable with those of the winning entry. However, the decision tree-based method is not as efficient as SPOT when dealing with data streams. The decision tree-based method is able to incrementally handle the data in the streams using the trees (or the rule sets) that are constructed in the learning stage, but it is not trivial to update the trees (or the rule sets) themselves incrementally as data are continuously arriving. Without updating

	Winning entry of KDD CUP'99	SPOT
Detection rate	91%	92.3%
False positive rate	0.55%	1.5%

Table 6.12: Comparing SPOT and the winning entry of KDD CUP'99

the trees (or the rule sets) in a real-time manner, it is impossible for this technique to capture the latest data characteristics and cope with possible concept drift in the streams. This is not desired in data stream applications. In contrast, SPOT is able to update PCS in a timely fashion each time when a new data from the stream is processed, enabling SPOT to handle the dynamics of data streams efficiently.

6.3 Summary

This section elaborates on the performance evaluation of SPOT. A fair amount of efforts have been taken for data preparation and interface development, an important work to facilitate the subsequent experimental evaluation. More specifically, we devise two synthetic data generators with difference emphasis. The data sets generated using the first data generator are general and well suited to evaluate the scalability of SPOT under varying number and dimensionality of data, while the data sets generated by the second generator are quite unique in terms of its data distribution. This helps to evaluate the detection effectiveness of SPOT and other competitive methods in some extreme cases. Application interfaces are also developed for the wireless network anomaly detection and the KDD-CUP'99 outlier detection applications to facilitate the application of SPOT. The major task in the wireless network anomaly detection is the generation of ground-truth results as the data set is unlabeled. For the KDD-CUP'99 outlier detection application, a number of issues, such as training data generation, redundant outlying subspace removal, anomaly categorization and false positive reduction, have been addressed.

A wide spectrum of experiments have been conducted in this section as well. The experimental results show that SPOT is scalable with respect to the length and dimensionality of data streams. They also show that SPOT features a good

	Detection rate	False positive rate
SPOT	1	1
Histogram	3	1
Kernel	1	1
Incremental LOF	1	1
HPStream	2	1

Table 6.13: Performance rank of different methods for data streams generated by SD2

	Detection rate	False positive rate
SPOT	1	3
Histogram	4	2
Kernel	3	4
Incremental LOF	2	1
HPStream	5	5

Table 6.14: Performance rank of different methods for KDD-CUP'99 data stream

convergence for outlying subspace search by using MOGA and the ability to handle data change in the data stream through self-evolution.

Comparative experiments carried in synthetic data sets, the wireless network anomaly detection and the KDD-CUP'99 outlier detection applications demonstrate that SPOT is effective in detecting anomalies. To obtain a big picture, we rank SPOT and different competitive methods based on their performance. Table 6.13, Table 6.14 and Table 6.15 present the ranking results of methods in the three different data streams. Better methods are assigned smaller ranks in these tables. Methods are assigned tie rank if their performance is close and not significantly distinguishable. For example in Figure 6.13, all the methods have the same rank in terms of false positive rate because all the methods have zero or close-to-zero false positive rate due to the simplicity of the data streams generated by SD2. From these three tables, we can see that SPOT performs best in terms of detection rate (for SD2 and KDD-CUP'99 data stream) and recall (for wireless network data stream). This is due to its strong search ability to explore subspaces for outlier detection. However, in terms of false positive rate and precision in the wireless network and KDD-CUP'99 applications, SPOT is not the best method. It is inferior to histogram method and Incremental

	Precision	Recall	F-measure
SPOT	3	1	1
Histogram	2	5	4
Kernel	4	2	3
Incremental LOF	1	3	2
HPStream	5	6	6
Largest_Cluster	6	4	5

Table 6.15: Performance rank of different methods for wireless network data stream

LOF. The histogram method has a high precision because it only explores a rather small number of subspaces. In other words, good precision is achieved in histogram method at a sacrifice of recall loss. Incremental LOF is accurate in identifying outliers due to use of a complicated outlier-ness metric (*i.e.*, LOF) that is computationally expensive. SPOT explores a remarkably larger number of subspaces than histogram method and uses simpler (yet much more efficient) outlier-ness metrics than Incremental LOF, therefore SPOT receives a higher pressure to produce false positives that adversely effects its detection precision. Despite this, SPOT still outperforms all the other methods in terms of F-measure for the wireless network data stream due to its much better recall performance.

Chapter 7

Conclusions

7.1 Research Summary

Even though the problem of outlier detection has been studied intensively in the past a few years, outlier detection problem in high-dimensional data streams has rarely been investigated. The current state-of-the-art methods of outlier detection are difficult in finding projected outliers in the context of data streams, because they are either constrained to only low dimensional data (relying on full dimensionality analysis), or not able to incrementally update detection model for real-time data streams. To the best of our knowledge, there has been little research work in literature that particularly targets this research problem.

This research work is motivated by the unique characteristics of data in high-dimensional space. Due to the curse of dimensionality, the outliers existing in the high-dimensional data sets (including data streams) are embedded in those lower dimensional subspaces. These outliers existing in high-dimensional data space are termed projected outliers. In this thesis, we introduce and formally formulate the problem of projected outlier detection for multi and high-dimensional data streams. To solve this problem, we present a new technique, called Stream Projected Outlier deTector (SPOT).

SPOT utilizes compact data synopsis, including BCS and the PCS, to capture necessary data statistical information for outlier detection. Both of them can be computed and maintained efficiently, enabling SPOT to meet the one-pass constraint and time criticality posed by data stream applications.

Given the inherent NP hardness of detecting projected outlier detection in high-dimensional data streams, we construct the Sparse Subspace Template (SST) in SPOT

to detect projected outliers. It is hoped that SST is able to contain the major subspaces where most of the projected outliers are embedded. In order to achieve this objective, SST consists of a number of mutually supplemented subspace groups that contribute collectively to an effective detection of projected outliers. The component of Fixed SST Subspaces ($\mathcal{FS}$) in SST covers the full space lattice whose dimensionality is upper bounded by $MaxDimension$. $\mathcal{FS}$ tries to establish the bottomline for detection performance of SPOT. Unsupervised SST Subspaces ($\mathcal{US}$) and Supervised SST Subspaces ($\mathcal{SS}$), being the other two components of SST, find supplemental subspaces that are not covered by $\mathcal{FS}$. Through $\mathcal{US}$ and $\mathcal{SS}$, SPOT is able to support both unsupervised and supervised learning to construct SST, providing a maximum level of flexibility to users.

Another major feature of SPOT lies in the outlying search strategy it uses to construct SST, particularly $\mathcal{US}$ and $\mathcal{SS}$. Unlike most of other outlier detection methods that measure outlier-ness of data points based on a single criterion, SPOT adopts a more flexible framework for using multiple measures for outlier detection. Employing multiple measures is generally more effective in measuring data sparsity than a single measure. In addition, SPOT utilizes Multi-Objective Genetic Algorithm (MOGA) as an effective search method to find subspaces that are able to optimize these outlier-ness criteria. We have presented in this thesis the ad-hoc design of MOGA in the aspects of objective/fitness functions, individual representation, selection operation, search operation, elitism and solution diversity preservation.

We have carried out a wide spectrum of experiments to evaluate the performance of SPOT in this thesis. Two synthetic data sets and four real-life data sets are used in the experimental evaluations. Two detailed case studies are conducted in the performance evaluation, namely MIT wireless network anomaly detection and KDD CUP'99 anomaly detection. Interfaces are developed to facilitate SPOT to be deployed in these applications by bridging the gap between the functionality of SPOT and the application-specific goals. The experimental results demonstrate that SPOT is efficient and effective in detecting projected outliers from high-dimensional data streams, and generally outperforms the related outlier detection methods using

histogram, Kernel function, density-based metric and clustering analysis.

7.2 Limitations of SPOT

Despite its advantages, SPOT still has some limitations. These limitations are summarized as follows.

- Because there are typically several thousands of subspaces in SST where SPOT needs to evaluate each data in the stream, thus it is not surprising to find that SPOT is noticeably slower than most of other traditional outlier detection methods that only evaluate the data in the full space or a single user-specified space. There exists a dilemma for us to specify the size of SST: a smaller SST will lead to a faster processing speed but has a higher chance of missing real outliers, and vice versa. Fortunately, most of outliers are embedded in the lower dimensional subspaces, therefore a reasonable SST (*e.g.*, $Maxidimension = 3$ and $|\mathcal{US}| + |\mathcal{SS}| \in [100, 500]$) can perform well. But, this is still largely a purely heuristic-based approach without any theoretical performance guarantee;

- Another difficulty in specifying a good SST size is that a large SST will impose a stronger pressure for SPOT to result in a high false positive rate, though the detection rate can be very high. This problem can be solved for SPOT, to a large extent, if labeled exemplars are available, based on which signature subspace lookup table can be built and false positive reduction can be performed. However, in absence of labeled exemplars, there have not been effective ways yet to automatically screen out those false positives for SPOT. The solution of this problem relies on a fair amount of domain knowledge for identifying the true outliers from the detected anomalies. As such, this problem cannot be solved by solely improving SPOT itself without resorting to external knowledge.

- SPOT is designed to detect point outliers from vector-type data sets. It is only able to deal with well structured data sets such as relational databases and data streams. It cannot be applied to other unstructured or semi-structured data sets such as TCP dump data or XML data.

7.3 Future Research

At the end of this subsection, we would like to lay out some potential research directions in outlier detection using SPOT for the future.

7.3.1 Adaptation of SST

In the current implementation of SPOT, we use a fixed-sized SST in the whole detection process. The size of SST remains unchanged once it has been constructed in the training stage. However, it is possible that some of the subspaces in SST, particularly in $\mathcal{FS}$, are non-outlying subspaces or some outlying subspaces become non-outlying subspaces as time evolves. A possible future research is to devise a mechanism to automatically adjust SST so that it only contains the outlying subspaces at any time. The advantages for doing this is two-fold. First, the size of SST can be greatly reduced. This implies that less computational and space overhead can be achieved. Second, this can help reduce the pressure for producing a large number of false positives when false positive categorization is impossible.

7.3.2 Optimization of Partition Size δ

An important parameter used in SPOT is δ, the number of intervals that each dimension of data is split into. This parameter determines cell granularity of the hypercube we create. For the reason of simplicity, we assume that each dimension is partitioned into an equal number of intervals with identical length. The specification of δ requires a close scrutiny. On one hand, the number of partition intervals should not be too small, otherwise the cells in the low dimensional subspaces generally cannot have statistically significant amount of data. Statistically sufficiency of data ensures the density and standard deviation of the populated base cells (based upon which RD, IRSD and IkRD are defined) serve as accurate measurements for data sparsity in the cell for outlier detection purpose. On the other hand, each cell needs to establish a reasonable neighborhood for data points, thus the partition cannot be too coarse either. Therefore, the optimization method for determining δ will be developed.

7.3.3 Distributed Outlier Detection Using SPOT

Most outlier detection methods, including SPOT, work in a single node mode. In other words, the outlier detector is working in an environment where the detector can process all the data streams under investigation. However, in some applications such as sensor networks, we need to deploy SPOT in a distributed mode, where SPOT are deployed to distributed sensors. In such case, each SPOT instance can only process the data streams that the sensor collects. It is required that the detection results of SPOT deployed in distributed nodes are identical to the results if SPOT is deployed a centralized node that can see the confluence of all the streams. The key challenge lies in that each distributed node needs to have a global, rather than a local, data synopsis the PCS in order to accurately detect global outliers using SPOT. We are interested in studying this problem in the future and developing a mechanism for efficient and effective outlier detection using SPOT in a distributed environment.

Bibliography

[1] C. C. Aggarwal. On Abnormality Detection in Spuriously Populated Data Streams. SIAM International Conference on Data Mining (SDM'05), Newport Beach, CA, 2005.

[2] B. Abraham and G. E. P. Box. Bayesian analysis of some outlier problems in time series. *Biometrika* 66, 2, 229-236, 1979.

[3] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein an J. Widom. STREAM: The Stanford Stream Data Manager, *SIGMOD'03*, 2003.

[4] B. Abraham and A. Chuang. Outlier detection and time series modeling. *Technometrics* 31, 2, 241-248, 1989.

[5] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation intrusion detection expert system (nides), software users manual, beta-update release. *Technical Report*, Computer Science Laboratory, SRI International, 1994.

[6] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. of *1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pp 94-105, 1998.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data Mining Application. In proceeding of *ACM SIGMOD'99*, Philadelphia, PA, USA, 1999.

[8] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A Framework for Clustering Evolving Data Streams. In Proc. of *29th Very Large Data Bases (VLDB'03)*,pp 81-92, Berlin, Germany, 2003.

[9] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In Proc. of *30th Very Large Data Bases (VLDB'04)*, pp 852-863, Toronto, Canada, 2004.

[10] F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In Proc. of *6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*,Helsinki, Finland, pp 15-26, 2002.

[11] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu and J. S. Park. Fast algorithms for projected clustering. In Proc. of *1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pp 61-72, 1999.

[12] D. Anderson, A. Tamaru, and A. Valdes. Detecting unusual program behavior using the statistical components of NIDES. *Technical Report*, Computer Science Laboratory, SRI International, 1995.

[13] C. C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pp 70-81, 2000.

[14] C. C. Aggarwal and P. S. Yu. Outlier Detection in High Dimensional Data. In Proc. of *2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, Santa Barbara, California, USA, 2001.

[15] Charu C. Aggarwal and Philip S. Yu. 2005. An effective and efficient algorithm for high-dimensional outlier detection. *VLDB Journal*, 14: 211-221, Springer-Verlag Publisher.

[16] Daniel Barbara. 2002. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, Volume 3, Issue 2, 23 - 27, ACM Press.

[17] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[18] V. Barnett. The ordering of multivariate data (with discussion). *Journal of the Royal Statistical Society*. Series A 139, 318-354, 1976.

[19] C. Bishop. Novelty detection and neural network validation. In Proceedings of *IEEE Vision, Image and Signal Processing*, Vol. 141. 217-222, 1994.

[20] M. Balazinska and P. Castro: Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. *MobiSys*, 2003.

[21] R. J. Beckman and R. D. Cook. Outliers. *Technometrics* 25, 2, 119-149, 1983.

[22] S. T. Brugger and J. Chow. An assessment of the DARPA IDS Evaluation Dataset using Snort. *Technical Report* CSE-2007-1, University of California, Davis, Department of Computer Science, Davis, CA, 2007.

[23] C. Bohm, C. Faloutsos and C. Plant. Outlier-robust clustering using independent components. *SIGMOD'08*, 185-198, 2008.

[24] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft. When is nearest neighbors meaningful? In Proc. of *7th International Conference on Database Theory (ICDT'99)*, pp 217-235, Jerusalem, Israel, 1999.

[25] M. M. Breunig, H-P Kriegel, R. T. Ng and J. Sander. OPTICS-OF: Identifying Local Outliers. *PKDD'99*, 262-270, 1999.

[26] M. Breuning, H-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, pp 93-104, 2000.

[27] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: an efficient and robust access method for points and rectangles. In Proc. of *1990 ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, pp 322-331, Atlantic City, NJ,1990.

[28] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley, 3rd edition, 1994.

[29] L. Boudjeloud and F. Poulet. Visual Interactive Evolutionary Algorithm for High Dimensional Data Clustering and Outlier Detection. In Proc. of *9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'05)*, Hanoi, Vietnam, pp426-431, 2005.

[30] Branch, J. Szymanski, B. Giannella, C. Ran Wolff Kargupta, H. n-Network Outlier Detection in Wireless Sensor Networks. In Proc. of. *26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[31] H. Cui. Online Outlier Detection Detection Over Data Streams. *Master thesis*, Simon Fraser University, 2002.

[32] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD'04*, pages 112-124, 2004.

[33] V. Chandola, A. Banerjee, and V. Kumar. Outlier Detection-A Survey, *Technical Report*, TR 07-017, Department of Computer Science and Engineering, University of Minnesota, 2007.

[34] C. Chow and D. Y. Yeung. Parzen-window network intrusion detectors. In Proceedings of *the 16th International Conference on Pattern Recognition*, Vol. 4, Washington, DC, USA, 40385, 2002.

[35] D. E. Denning. An intrusion detection model. *IEEE Transactions of Software Engineering* 13, 2, 222-232, 1987.

[36] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature* (PPSN VI), pages 849-858, Berlin, 2000.

[37] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. *International Conference on Machine Learning (ICML'06)*, 233-240, 2006.

[38] M. Desforges, P. Jacob, and J. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. In Proceedings of *Institute of Mechanical Engineers*, Vol. 212. 687-703, 1998.

[39] D. Dasgupta and F. Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In Proceedings of *the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1. Nashville, TN, 125-130, 2000.

[40] K. Das and J. G. Schneider: Detecting anomalous records in categorical datasets. *KDD'07*, 220-229, 2007.

[41] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In Proceedings of *the Seventeenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc., 2000.

[42] D. Endler. Intrusion detection: Applying machine learning to solaris audit data. In Proceedings of the 1*4th Annual Computer Security Applications Conference*, 268, 1998.

[43] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. *Applications of Data Mining in Computer Security*, 2002.

[44] M. Ester, H-P Kriegel, J. Sander, and X.Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In proceedings of *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Portland, Oregon, USA, 1996.

[45] E. Eskinand and S. Stolfo. Modeling system call for intrusion detection using dynamic window sizes. In Proceedings of *DARPA Information Survivability Conference and Exposition*, 2001.

[46] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, Series B (Methodological) 34, 3, 350-363, 1972.

[47] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In Proceedings of *International Conference on Genetic Algorithms and Their Applications*, pages 141-153, Pittsburgh, PA, 1985.

[48] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Proceedings of *the Fifth International Conference on Genetic Algorithms*, pages 416-423, San Mateo, California, 1993.

[49] T. Fawcett. and F. Provost. Activity monitoring: noticing interesting changes in behavior. In Proceedings of the 5th *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 53-62, 1999.

[50] D. Freedman, R. Pisani and R. Purves. *Statistics*, W. W. Norton, New York, 1978.

[51] F. Grubbs Procedures for detecting outlying observations in samples. *Technometrics* 11, 1, 1-21, 1969.

[52] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[53] S. Guttormsson, R. M. II, and M. El-Sharkawi. Elliptical novelty grouping for on-line short-turn detection of excited running rotors. *IEEE Transactions on Energy Conversion* 14, 1, 1999.

[54] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering Large Datasets in Arbitrary Metric Spaces. In Proc.s of *the 15th International Conference on Data Engineering (ICDE'99)*, Sydney, Australia, 1999.

[55] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In Proceedings of *the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, WA, USA, 1998.

[56] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor MI, 1975.

[57] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[58] P. Helman and J. Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. In *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 27, 449-466, 1997.

[59] P. S. Horn, L. Feng, Y. Li, and A. J. Pesce. Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry* 47, 12, 2137-2145, 2001.

[60] J. Han and M Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2000.

[61] A. Hinneburg, and D.A. Keim. An Efficient Approach to Cluster in Large Multimedia Databases with Noise. *KDD'98*, 1998.

[62] P. Hajela and C.-Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99-107, 1992.

[63] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In Proceedings of *the First IEEE Conference on Evolutionary Computation*, volume 1, pages 82-87, Piscataway, NJ, 1994. IEEE Press.

[64] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In Proceedings of *1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 119-124, Piscataway, NJ, 1996. IEEE Press.

[65] K. A. De Jong. Analysis of the Behavior of a Class of Genetic Adaptive Systems. *Ph. D. Dissertation*, University of Michigan, Ann Arbor, MI, 1975.

[66] W. Jin, A. K. H. Tung and J. Han. Finding Top n Local Outliers in Large Database. In Proc. of *7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'01)*, San Francisco, CA, pp 293-298, 2001.

[67] W. Jin, A. K. H. Tung, J. Han and W. Wang: Ranking Outliers Using Symmetric Neighborhood Relationship. *PAKDD'06*, 577-593, 2006.

[68] H. S. Javitz and A. Valdes. The SRI IDES statistical anomaly detector. In Proceedings of *the 1991 IEEE Symposium on Research in Security and Privacy*, 1991.

[69] F. Kursawe. A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature*, page 193-197, Berlin, 1991. Springer.

[70] J. D. Knowles and D. W. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Congress on Evolutionary Computation (CEC99)*, volume 1, pages 98-105, Piscataway, NJ, 1999.

[71] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimization by Simulated Annealing. *Science* (220) (4589): pages 671-680, 1983.

[72] G. Karypis, E-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer*, 32, Pages 68-75, 1999.

[73] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In Proceedings of *the 19th Annual Computer Security Applications Conference*, 14, 2003.

[74] E. M. Knorr and R. T. Ng. A unified approach for mining outliers. *CASCON'97*, 11, 1997.

[75] E. M. Knorr and R. T. Ng. A Unified Notion of Outliers: Properties and Computation. *KDD'97*, 219-222, 1997.

[76] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-based Outliers in Large Dataset. In Proc. of *24th International Conference on Very Large Data Bases (VLDB'98)*, New York, NY, pp 392-403, 1998.

[77] E. M. Knorr and R. T. Ng (1999). Finding Intentional Knowledge of Distance-based Outliers. In Proc. of *25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, pp 211-222, 1999.

[78] E. M. Knorr, R. T. Ng and V. Tucakov. Distance-Based Outliers: Algorithms and Applications. *VLDB Journal*, 8(3-4): 237-253, 2000.

[79] T. M. Khoshgoftaar, S. V. Nath, and S. Zhong. Intrusion Detection in Wireless Networks using Clusterings Techniques with Expert Analysis. Proceedings of *the Fourth International Conference on Machine Leaning and Applications (ICMLA'05)*, Los Angeles, CA, USA, 2005.

[80] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John wiley&Sons, 1990.

[81] C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In Proceedings of *the 2002 ACM Symposium on Applied computing*, 201-208, 2002.

[82] X. Li and J. Han: Mining Approximate Top-K Subspace Anomalies in Multi-Dimensional Time-Series Data. *VLDB*, 447-458, 2007.

[83] J. Lee, J. Han and X. Li. Trajectory Outlier Detection: A Partition-and-Detect Framework. *ICDE'08*, 140-149, 2008.

[84] J. Laurikkala, M. Juhola1, and E. Kentala. 2000. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 20-24, 2000.

[85] G. Manson. Identifying damage sensitive, environment insensitive features for damage detection. In Proceedings of *the IES Conference*. Swansea, UK, 2002.

[86] J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proc. of *5th Berkeley Symp. Math. Statist, Prob.*, 1, pages 281-297, 1967.

[87] M. V. Mahoney and P. K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In Proceedings of *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 376-385, 2002.

[88] G. Manson, G. Pierce, and K. Worden. On the long-term stability of normal condition for damage detection in a composite panel. In Proceedings of *the 4th International Conference on Damage Assessment of Structures*, Cardiff, UK, 2001.

[89] G. Manson, S. G. Pierce, K. Worden, T. Monnier, P. Guy, and K. Atherton. Long-term stability of normal condition data for novelty detection. In Proceedings of *Smart Structures and Integrated Systems*, 323-334, 2000.

[90] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD'03*, pages 631-636, 2003.

[91] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In proceedings of *the 20th VLDB Conference*, pages 144-155, 1994.

[92] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-Data Algorithms For High-Quality Clustering. In Proceedings of *the 18th International Conference on Data Engineering (ICDE'02)*, San Jose, California, USA, 2002.

[93] M. I. Petrovskiy. Outlier Detection Algorithms in Data Mining Systems. *Programming and Computer Software*, Vol. 29, No. 4, pp 228-237, 2003.

[94] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics* 33, 1065-1076, 1962.

[95] S. Papadimitriou, H. Kitagawa, P. B. Gibbons and C. Faloutsos: LOCI: Fast Outlier Detection Using the Local Correlation Integral. *ICDE'03*, 315, 2003.

[96] S. Papadimitriou and C. Faloutsos. Cross-Outlier Detection. *SSTD'03*, 199-213, 2003.

[97] D. Pokrajac, A. Lazarevic, L. Latecki. Incremental Local Outlier Detection for Data Streams, *IEEE symposiums on computational Intelligence and Data Mining (CIDM'07)*, 504-515, Honolulu, Hawaii, USA, 2007.

[98] G. T. Parks and I. Miller. Selective breeding in a multiobjective genetic algorithm. *Parallel Problem Solving from Nature*, PPSN V, pages 250-259, Berlin, 1998.

[99] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In Proceedings of *20th NIST-NCSC National Information Systems Security Conference*, 353-365, 1997.

[100] T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Record* 32(4): 77-82, 2003.

[101] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, pp 427-438, 2000.

[102] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In Proceedings of *International Conference on Genetic Algorithms and Their Applications*, pages 93-100, Pittsburgh, PA, 1985.

[103] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, London, 1986.

[104] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Wavelet based Clustering Approach for Spatial Data in Very Large Database. *VLDB Journal*, vol.8 (3-4), pages 289-304, 1999.

[105] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221-248, 1994.

[106] H. E. Solberg and A. Lahti. Detection of outliers in reference distributions: Performance of horn's algorithm. *Clinical Chemistry* 51, 12, 2326-2332, 2005.

[107] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos. Neighborhood Formation and Anomaly Detection in Bipartite Graphs. *ICDM'05*, 418-425, 2005.

[108] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos. Relevance search and anomaly detection in bipartite graphs. *SIGKDD Explorations* 7(2): 48-55, 2005.

[109] I. A. Sarafis and P. W. Trinder and A. M. S. Zalzala. Towards Effective Subspace Clustering with an Evolutionary Algorithm. In Proc. of *2003 IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003.

[110] L. Tarassenko. Novelty detection for the identification of masses in mammograms. In Proceedings of *the 4th IEEE International Conference on Artificial Neural Networks*, Vol. 4. Cambridge, UK, 442-447, 1995.

[111] J. Tang, Z. Chen, A. Fu, and D. W. Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In Proc. of *6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*, Taipei, Taiwan, 2002.

[112] T. B Tack, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3-17, 1997.

[113] W. Wang, J. Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In Proceedings of *23rd VLDB Conference*, pages 186-195, Athens, Green, 1997.

[114] W. Wang, J. Zhang and H. Wang. Grid-ODF: Detecting Outliers Effectively and Efficiently in Large Multi-dimensional Databases. In Proc. of *2005 International Conference on Computational Intelligence and Security (CIS'05)*, pp 765-770, Xi'an, China, 2005.

[115] K. Yamanishi and J. I. Takeuchi. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In Proceedings of *the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 389-394, 2001.

[116] K. Yamanishi, J. Takeichi, and G. Williams. On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. In Proc. of *2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Boston, pp. 320-324, 2000.

[117] K. Yamanishi, J. I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery* 8, 275-300, 2004.

[118] C.T. Zahn. Graph-theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transaction on Computing*, C-20, pages 68-86, 1971.

[119] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173-195, 2000.

[120] J. Zhang, Q. Gao and H. Wang. Outlying Subspace Detection in High dimensional Space. *Encyclopedia of Database Technologies and Applications (2nd Edition)*, Idea Group Publisher, 2009.

[121] J. Zhang and H. Wang. Detecting Outlying Subspaces for High-dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems: An International Journal (KAIS)*, Springer-Verlag Publisher, 2006.

[122] J. Zhang, W. Hsu and M. L. Lee. Clustering in Dynamic Spatial Databases. *Journal of Intelligent Information Systems (JIIS)* 24(1): 5-27, Kluwer Academic Publisher, 2005.

[123] C. Zhu, H. Kitagawa and C. Faloutsos. Example-Based Robust Outlier Detection in High Dimensional Datasets. In Proc. of *2005 IEEE International Conference on Data Management(ICDM'05)*, pp 829-832, 2005.

[124] C. Zhu, H. Kitagawa, S. Papadimitriou and C. Faloutsos. OBE: Outlier by Example. *PAKDD'04*, 222-234, 2004.

[125] S. Zhong, T. M. Khoshgoftaar, and S. V. Nath. A clustering approach to wireless network intrusion detection. In *ICTAI*, pages 190-196, 2005.

[126] J. Zhang, M. Lou, T. W. Ling and H. Wang. HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data. In Proc. of *30th International Conference on Very Large Data Bases (VLDB'04)*, demo, pages 1265-1268, Toronto, Canada, 2004.

[127] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. *Evolutionary Methods for Design, Optimisation, and Control*, pages 19-26, Barcelona, Spain, 2002.

[128] J. Zhang, Q. Gao and H. Wang. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *2006 IEEE International Conference on Data Mining (ICDM'06)*, pages 731-740, Hong Kong, China, 2006.

[129] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In proceedings of *the 1996 ACM International Conference on Management of Data (SIGMOD'96)*, pages 103-114, Montreal, Canada, 1996.

[130] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257-271, 1999.

[131] J. Zhang and H. Wang. 2006. Detecting Outlying Subspaces for High-dimensional Data: the New Task, Algorithms and Performance. *Knowledge and Information Systems (KAIS)*, 333-355, 2006.